Sistemas informáticos
Curso 2010/2011

Proyecto

Sistema de extensiones para la generación de videojuegos educativos

Autores:

Nuria García Santa

Alicia Pérez Jiménez

Francisco Javier Laguna García

Dirigido por:

Pilar Sancho Thomas

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

Palabras clave para búsqueda bibliográfica:

- e-Learning
- Videojuego
- Educativo
- Plugin
- e-Adventure
- Jin-Plugin
- Google Guice
- Plataforma

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado.

Alicia Pérez Jiménez Nuria García Santa Francisco Javier Laguna García

Resumen

El proyecto desarrollado consiste en la realización de una plataforma de gestión de plugins para la herramienta de creación de videojuegos educativos e-Adventure, desarrollada en la Universidad Complutense de Madrid. Este sistema permite separar el núcleo de la aplicación e-Adventure de las posibles extensiones que se le puedan incorporar, facilitando su mantenimiento y escalabilidad. Además, la introducción de un sistema de plugins en e-Adventure permitirá aumentar el tipo de juegos que pueden generarse con la plataforma. Para ello, se ha realizado un estudio detallado de las herramientas de creación de videojuegos, así como de diferentes gestores de plugins que podemos encontrar actualmente.

Además, se procede a la implementación de una serie de plugins de ejemplo para probar toda la funcionalidad del gestor.

Abstract

The project developed consists of the implementation of a platform for plugin management that has been integrated into the educational game creation tool e-Adventure. e-Adventure has been developed at the Complutense University of Madrid. This system splits apart the core application of e-Adventure and the possible extensions that it can be incorporated, facilitating maintenance and scalability. By introducing a plugin architecture in the platform it will be easier in the future to develop new kinds of games with e-Adventure. To achieve this goal, a detailed study of videogames creation tools has been conducted, and different plugins have been developed as examples of the potential of the system.

Agradecimientos

Este trabajo ha sido fruto de una gran dedicación y esfuerzo. Estamos muy satisfechos por el resultado obtenido.

En primer lugar, queremos agradecer a nuestra directora de proyecto, Pilar Sancho Thomas, por su apoyo y ayuda durante la realización del mismo.

En segundo lugar, pero no por ello menos importante, agradecemos al equipo de desarrollo de e-Adventure toda su ayuda, paciencia y tiempo. En especial a Eugenio J. Marchiori, Ángel del Blanco Aguado y Francisco Javier Torrente Vigil.

¡Gracias por aguantarnos todo este año! Hemos aprendido mucho gracias a vosotros.

Tabla de contenidos

Capítulo I
Introducción

Tabla de contenidos

1. Introducción

1.1 Plataforma e-Adventure[1]

El desarrollo de videojuegos es una tarea a tiempo completo que requiere conocimientos técnicos avanzados y de programación. Muy pocos proyectos educativos podrían afrontar el gasto de un videojuego. Con la idea de solventar estas limitaciones surgió el proyecto e-Adventure. De esta forma, se desarrolló una plataforma intuitiva pero completa y potente para el desarrollo de aventuras con fines educativos y a un coste razonable.

La plataforma e-Adventure es un proyecto de investigación que aspira a facilitar la integración de juegos educativos y simulaciones basadas en juego en procesos educativos en general y Entornos de Aprendizaje Virtuales (VLE) en particular.

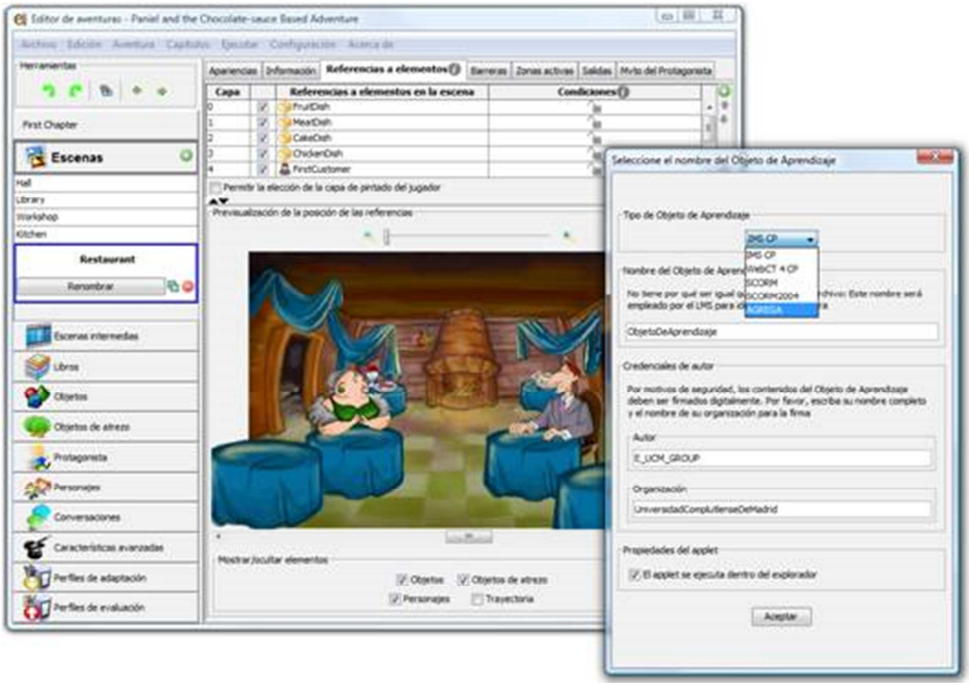

Figura 1. Vista de editor de juegos de e-Adventure

1 e-Adventure http://e-adventure.e-ucm.es

Actualmente, la investigación del proyecto se centra en tres áreas específicas:

- Reducción de los costes de desarrollo para videojuegos educativos.
- Incorporación de características educativas específicas en herramientas de desarrollo de juegos.
- Integración de los juegos resultantes con cursos existentes en Entornos de Aprendizaje Virtuales.

La plataforma consta de dos aplicaciones, la herramienta de edición orientada a autor y el motor de juego. El diseñador del juego toma los recursos producidos por los artistas y desarrolla el juego con el editor. Por su lado el motor de juego es el encargado de ejecutar dichos juegos. Los educadores no necesitan tener ningún conocimiento técnico sobre el desarrollo de videojuegos, simplemente han de centrarse en los aspectos educativos.

1.2 Extensión de la plataforma e-Adventure

Durante estos años, e-Adventure ha conseguido una alta aceptación y un gran número de usuarios (Figura 2). Al tratarse de una aplicación enfocada a una funcionalidad concreta, se hace más complicada la tarea de adaptarla a nuevos requisitos y funcionalidades. Esto ha supuesto algunas limitaciones en la herramienta, debido a la dificultad para aumentar sus prestaciones extendiendo las posibilidades que ofrece. También aumenta el trabajo de mantenimiento de la plataforma tras experimentar un gran crecimiento desde sus características iniciales.

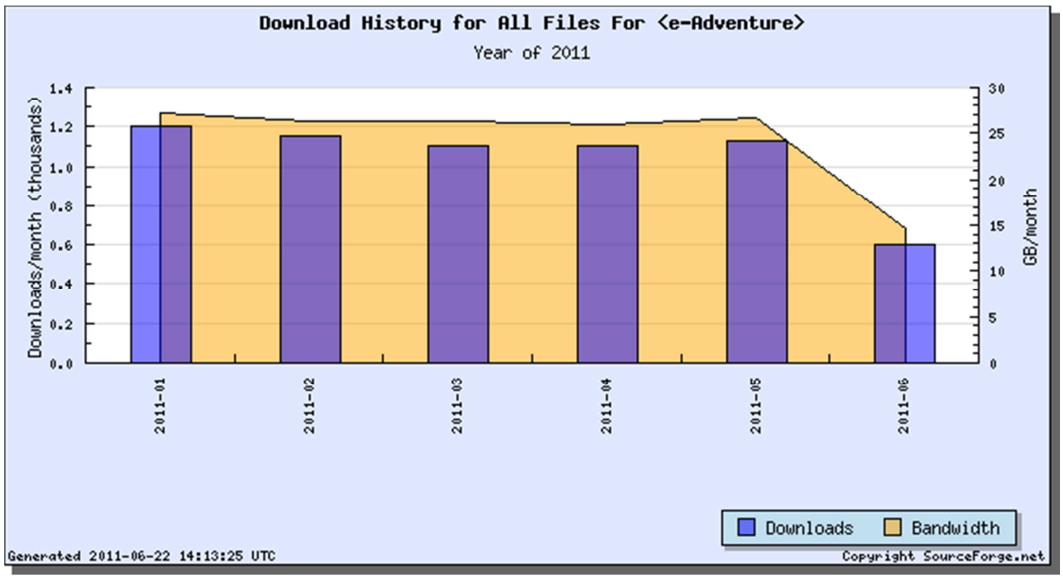

Figura 2. Descargas de la plataforma e-Adventure durante el año 2011

Por estos motivos, ha surgido la idea de este proyecto, con el cual se quieren resolver las limitaciones de extensión de la plataforma, y los problemas que ello conlleva. Por tanto, el objetivo de nuestro proyecto se

centra en generalizar la herramienta de creación de videojuegos educativos para ampliar el tipo de juegos que pueden desarrollarse con la plataforma, ahora mismo limitado a aventuras gráficas 2D tipo point-and-click.

La idea central es lograr total independencia del núcleo principal de la aplicación, con el resto de funcionalidades incorporadas. De esta forma, se pretende establecer una abstracción de la lógica principal para que cualquier desarrollador ajeno a la plataforma pueda desarrollar sus propias extensiones, sin necesidad de conocer el funcionamiento interno de la herramienta.

De esta manera, el trabajo que se va a desarrollar en este proyecto dotará de un sistema de extensiones (plugins) que permitirá extender la capacidad expresiva de la plataforma e-Adventure mediante programación en Java.

Por último, destacar que este proyecto tiene un gran potencial, no sólo por la versatilidad que la incorporación de un sistema de extensiones añade a la propia herramienta, sino también por las propias extensiones de ejemplo a realizar, que permitirán explorar nuevas tecnologías, metáforas de juego, etc. dentro del mismo proyecto.

1.3 Sobre este documento

Procederemos a comentar brevemente la estructuración que se ha seguido en la documentación de nuestro proyecto.

Este documento se encuentra organizado en capítulos, planteados de forma cronológica. En los primeros capítulos se expone una introducción del proyecto, para acercar al lector a la temática de nuestro trabajo. El presente capítulo resume las ideas del proyecto. En los dos siguientes se realiza una descripción del estudio del dominio y del estado del arte. Posteriormente se presenta un análisis de los patrones y tecnologías empleadas. En los últimos capítulos se describe el diseño de nuestra aplicación y de los distintos casos de estudio desarrollados. Se terminará la documentación exponiendo las conclusiones sobre el proyecto.

Los contenidos redactados en la documentación son bastante diversos y completos, presentando tanto aspectos técnicos, como numerosas descripciones y capturas de pantalla. En las últimas páginas se mostrará una completa bibliografía, así como todo tipo de referencias utilizadas para el desarrollo del proyecto.

Capítulo II
Motivación y objetivos

Tabla de contenidos

2. Motivación y objetivos

En este apartado se va a proceder, primero a un resumen sobre la situación de los videojuegos, especialmente los educativos, a fecha actual, centrándonos en el ámbito tecnológico, económico y educativo.

En la segunda parte se explicarán los motivos principales por los cuales surge el proyecto. Básicamente, e-Adventure creció mucho pero sin la flexibilidad necesaria para un proyecto cada vez mayor. También se procederá a la descripción de los objetivos que el proyecto abarcará.

2.1 Introducción

Los videojuegos han evolucionado sin cesar en los últimos años. Han experimentado un gran crecimiento a nivel económico, tecnológico y social. La variedad en sus temáticas y su perfil de entretenimiento y ocio han contribuido a su expansión. La sociedad continúa descubriendo que los videojuegos son actividades que se pueden desarrollar en familia o entre amigos, y que tienen potencial como herramienta educativa.

Los videojuegos están tomando fuerza en su uso pedagógico y educativo. Su complejidad permite desarrollar habilidades para la resolución de problemas, la toma de decisiones, la búsqueda de información, la organización, etc. A partir de esto ha surgido el concepto de 'serious games', definición para calificar aquellos juegos que tienen objetivos educativos además de la diversión.

El principal problema de usar los videojuegos en educación (donde los presupuestos no son excesivamente elevados) radica en que son costosos de crear. Necesitan de una gran inversión económica y tecnológica. Para solucionar esto han surgido distintas aplicaciones que ayudan en su creación, como por ejemplo las herramientas de autoría, pero estas aplicaciones plantean otros problemas adicionales como es su dificultad de extensión, ya

que el abanico de posibilidades a la hora de crear un videojuego educativo es muy grande y la funcionalidad de estas herramientas suele ser limitada.

La ingeniería de software establece las pautas de análisis y diseño para facilitar la construcción y mantenimiento posterior de los sistemas software. De esta forma, favorece un desarrollo óptimo de las aplicaciones.

Cuando los sistemas se van ampliando y extendiendo aumenta el problema del mantenimiento, que se hace cada vez más difícil, y de extensión de funcionalidades no previstas en el diseño inicial. Por esto, la modularidad y dependencia de las partes de un sistema plantea una gran importancia en este aspecto. Una solución satisfactoria para lograr una buena modularidad es el desarrollo de un sistema con un núcleo común que permita la extensión del sistema original mediante plugins, que son extensiones independientes con funcionalidad concreta.

2.2 Estudio del dominio

2.2.1 Videojuegos en la actualidad

Desde sus comienzos en los años 70, los videojuegos han evolucionado incesantemente. Actualmente gozan de un gran crecimiento, no solo en el ámbito económico sino que también en el tecnológico, y aceptación social, lo que les ha permitido situarse en la cima del ocio y del entretenimiento.

La gran diversidad temática de los videojuegos ha contribuido a su expansión. Los juegos se han conseguido vincular a la realidad social, puesto que abarcan numerosas facetas de la vida. La forma de juego actual les ha ayudado a llegar a todo el mundo. La sociedad sigue descubriendo que los videojuegos aportan actividades para desarrollar en familia o entre amigos, y tienen potencial como herramienta educativa. También ha cambiado el perfil de los jugadores. El perfil del usuario medio es el de un varón de en torno a 20 años, que juega unas tres horas a la semana, generalmente solo y en su propia casa. Sin embargo, existen rotundas perspectivas de crecimiento en mercados como juegos online, móviles, y juegos para mujeres o personas mayores. En España, el 22,5% de la población se declara jugadora de videojuegos; en total, cerca de 10,4 millones de personas, del total de 46 millones de españoles[2]. Otro aspecto fundamental para su enriquecimiento social ha venido de la mano de los nuevos avances tecnológicos, que permiten al jugador adentrarse en la historia con un nivel de profundidad mucho mayor. Esto ha sido uno de los principales motivos de su gran éxito desde el punto de vista social.

Tecnológicamente, se ha avanzado de manera exponencial, ofreciendo al usuario mucha más interactividad en el juego, haciendo que se involucre y participe en todo momento. Con los nuevos dispositivos y hardware, se logra

[2] Anuario 2009 aDeSe (Asociación Española de distribuidores y Editores de Software de Entretenimiento) http://www.adese.es/pdf/Anuario2009aDeSe.pdf

activar todos los sentidos de los jugadores, así como su capacidad física y mental. Recientemente, se están imponiendo los videojuegos online, que necesitan de importantes infraestructuras y plataformas. Casi la mitad de los españoles que juegan en el PC lo hacen de manera online (Asociación Española de distribuidores y Editores de Software de Entretenimiento (aDeSe), Anuario 2009). Se facilita un gráfico (Figura 3) comparativo de los resultados de un estudio sobre ello.

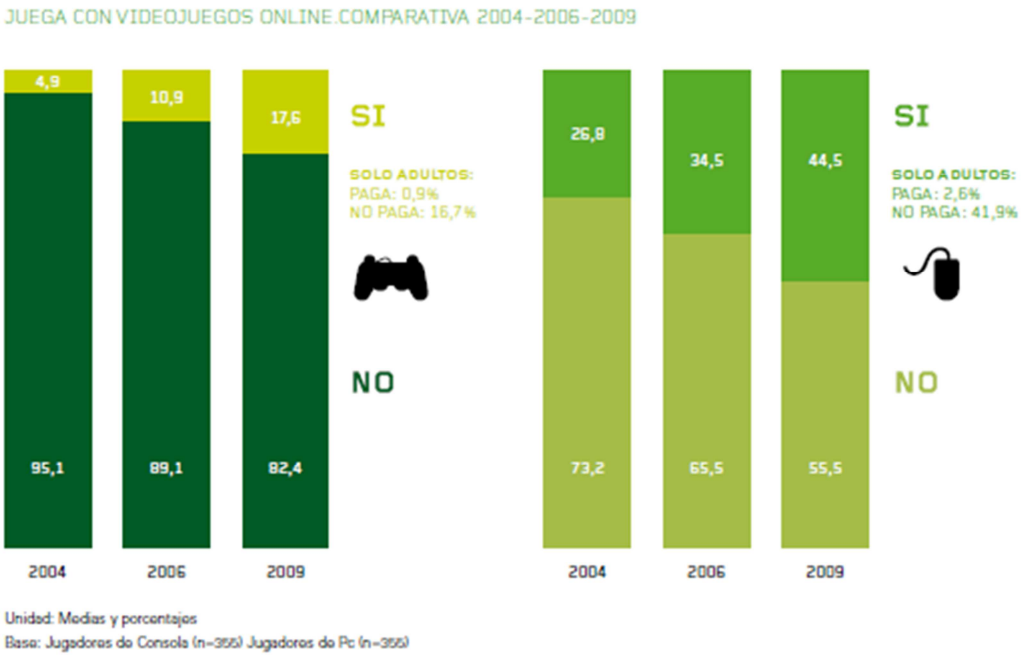

Figura 3. Tabla comparativa de videojuegos online de los años 2004-2006-2009(aDeSe, Anuario 2009)

El aumento de la popularidad de los videojuegos ha supuesto grandes beneficios económicos para las empresas del sector. Pero el problema de la piratería, que está alcanzando proporciones alarmantes en España, con un índice en piratería informática del 42%, está causando daños irreparables (aDeSe, Anuario 2009). A pesar de este hecho, nos encontramos en una época, en la que la industria del videojuego cuenta con un gran éxito de consumo audiovisual en comparación con la industria musical y cinematográfica (Figura 4).

En el último punto de este análisis sobre la situación actual de los videojuegos, nos centraremos en aquellos que cuentan con una temática educativa.

Hasta finales del siglo XIX, la acción de jugar había estado asociada al entretenimiento y a la diversión. Desde el punto de vista educativo, este hecho cambió gracias al movimiento pedagógico de la Escuela Nueva en la que el juego adquirió un importante protagonismo como metodología de enseñanza. Hoy en día, está ampliamente aceptado que se puede aprender jugando ([2]).

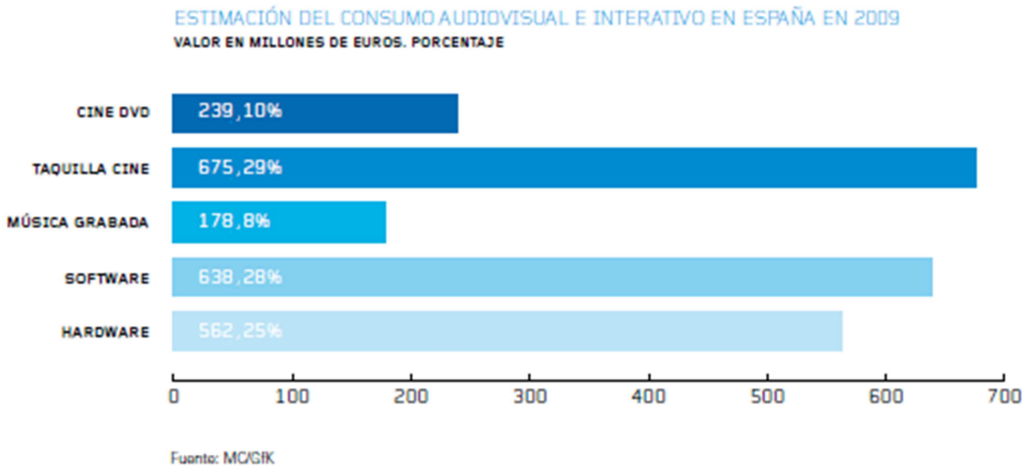

Figura 4. Estimación del consumo audiovisual e interactivo en España en 2009 (aDeSe, Anuario 2009)

Con el desarrollo de la tecnología informática surge un nuevo tipo de juego: los videojuegos. Su complejidad permite desarrollar habilidades para la resolución de problemas, la toma de decisiones, la búsqueda de información, la organización, etc ([2]).

Un concepto asociado que está tomando fuerza últimamente es el de los 'serious games', definición para calificar aquellos juegos que tienen objetivos educativos además de la diversión. Este nombre hace referencia a videojuegos que tratan sobre temas educativos, investigación científica, simuladores e ingeniería, entre otros. Se agrupan todos los juegos que faciliten el aprendizaje en distintos ámbitos de la educación.

También se utilizan con fines comerciales, de concienciación o de denuncia social o política. Pueden ayudar a la comprensión de conceptos o materias y como medio de comunicación o publicitario. Sirven igualmente como entrenamiento y simulación de distintas actividades, como por ejemplo simular el pilotaje de un avión.

A continuación, se muestra una tabla (Figura 5) de una posible forma de clasificar este tipo de juegos.

Se espera que el sector de los serious games crezca significativamente en los próximos años. Según un blog dedicado al tema de los serious games, se estima que actualmente genera unos 1.5 mil millones de euros de ingresos alrededor del mundo, y que para 2015 las ventas serán siete veces mayores de lo que son en 2010, con un crecimiento medio anual de un 47% entre 2010 y

2015. También se considera que las empresas incrementarán su interés por este tipo de juegos alrededor del 2013.[3]

	Games for Health	Advergames	Games for Training	Games for Education	Games for Science and Research	Production	Games as Work
Government & NGO	Public Health Education & Mass Casualty Response	Political Games	Employee Training	Inform Public	Data Collection / Planning	Strategic & Policy Planning	Public Diplomacy, Opinion Research
Defonso	Rehabilitation & Wellness	Recruitment & Propaganda	Soldier/Support Training	School House Education	Wargames / planning	War planning & weapons research	Command & Control
Healthcare	Cybertherapy / Exergaming	Public Health Policy & Social Awareness Campaigns	Training Games for Health Professionals	Games for Patient Education and Disease Management	Visualization & Epidemiology	Biotech manufacturing & design	Public Health Response Planning & Logistics
Marketing & Communications	Advertising Treatment	Advertising, marketing with games, product placement	Product Use	Product Information	Opinion Research	Machinima	Opinion Research
Education	Inform about diseases/risks	Social Issue Games	Train teachers / Train workforce skills	Learning	Computer Science & Recruitment	P2P Learning Constructivism Documentary?	Teaching Distance Learning
Corporate	Employee Health Information & Wellness	Customer Education & Awareness	Employee Training	Continuing Education & Certification	Advertising / visualization	Strategic Planning	Command & Control
Industry	Occupational Safety	Sales & Recruitment	Employee Training	Workforce Education	Process Optimization Simulation	Nano/Bio-tech Design	Command & Control

Figura 5. Tabla de una posible forma de clasificación de 'serious games'[4]

Algunos ejemplos de estos juegos son[5]:

- America's Army[6]: Diversas pruebas que tratan sobre el entrenamiento militar.
- Emergencia 112[7]: Enseña cómo se deben realizar primeros auxilios de forma adecuada.
- FoodForce[8]: Juego que instruye sobre la ayuda humanitaria.
- Catechumen[9]: Se trata de un juego religioso.
- Los famosos 'Brain Training'[10]: Método de entrenamiento mental cuyo objetivo es mantener el cerebro joven.

3 IDATE: Serious Games - A 10 Billion Euro Market In 2015 Serious Games Market http://seriousgamesmarket.blogspot.com/2010/08/idate-serious-games-10-billion-euro.html
4 Richard Carey. Taxonomía de juegos serios (serious games). http://www.richardcarey.net
5 José Vicente Pons Alfonso - ¿Qué son los "Serious Games" (juegos serios)? http://www.exelweiss.com/blog/356/serious-games-juegos-serios
6 Juego "America's Army" http://www.americasarmy.com
7 Juego "Emergencia 112" http://www.exelweiss.com/desarrollo/videojuegos/movil/java/emergencia112.php
8 Juego "Food force" http://www.wfp.org/how-to-help/individuals/food-force
9 Juego "Catechumen" http://www.n-lightning.com/catechumen.htm
10 Serie de juegos "Brain Training" de Nintendo http://www.nintendo.es

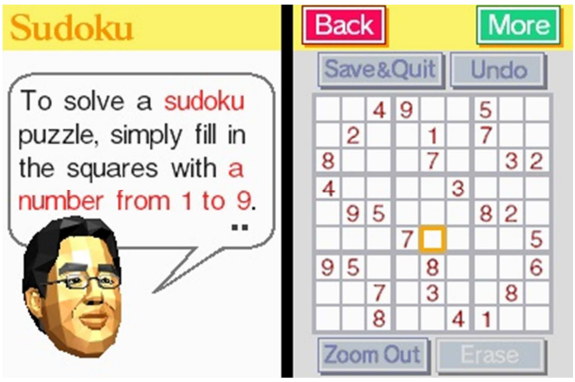

Figura 6. Imagen de ejemplo de juego tipo "Brain Training"[11]

La creación de videojuegos presenta algunas dificultades, como el elevado coste de su realización o la necesidad de personal multidisciplinar (programadores, diseñadores, guionistas…). Con el objetivo de solventar estos problemas han surgido en los últimos años herramientas que facilitan el desarrollo de videojuegos reduciendo las habilidades técnicas necesarias para llevarlos a cabo. Esto permite ampliar el perfil de personas que pueden realizar un videojuego, y simplifica la construcción de los mismos, reduciendo a su vez las funcionalidades del juego a aquellas contempladas en la plataforma.

2.2.2 Ingeniería del software y modularidad

La Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como Desarrollo de Software o Producción de Software (Bohem, 1976). Entre sus objetivos destaca la producción de software con calidad.

Calidad implica dos tipos de factores: internos y externos. Los factores externos son las cualidades que son "detectadas" por los usuarios, por ejemplo, la velocidad y la facilidad de uso; los factores internos son cualidades perceptibles por profesionales del área de la computación, por ejemplo, la modularidad y la legibilidad.

Algunos factores externos:

- Correctitud: Capacidad para realizar con exactitud las tareas definidas en las especificaciones.
- Robustez: Capacidad de reaccionar apropiadamente ante condiciones excepcionales.
- Extensibilidad: Facilidad de adaptar los productos de software a los cambios en la especificación.
- Reutilización: Capacidad de los elementos de software de servir para la construcción de múltiples aplicaciones diferentes.

[11] Imagen de “Brain Tranining” http://mundogeek.net

- Compatibilidad: Facilidad de combinar unos elementos de software con otros.
- Eficiencia: Capacidad para exigir la menor cantidad posible de recursos (tiempo de procesador, espacio de memoria, ancho de banda, etc.).
- Portabilidad: Facilidad de transferir los productos de software a diferentes entornos de hardware y software.
- Facilidad de uso: Cubre la facilidad desinstalación, de operación y de supervisión.
- Funcionalidad: Conjunto de posibilidades que proporciona un sistema.

El mantenimiento no se menciona como un factor, pero se estima que gran parte del costo del software se dedica al mismo.

A partir de los objetivos de extensibilidad y reutilización, dos de los factores de calidad más importantes, se desprende la necesidad de tener arquitecturas de sistemas flexibles, hechas con componentes autónomos de software, y esto se logra con una adecuada modularidad.

Modularidad software

Es la capacidad que tiene un sistema de ser estudiado, visto o entendido como la unión de varias partes que interactúan entre sí y que trabajan para alcanzar un objetivo común, realizando cada una de ellas una tarea necesaria para la consecución de dicho objetivo. Cada una de esas partes en que se encuentre dividido el sistema recibe el nombre de módulo, y de forma ideal un módulo debe poder cumplir las condiciones de caja negra, es decir, ser independiente del resto de los módulos y comunicarse con ellos (con todos o sólo con una parte) a través de unas entradas y salidas bien definidas.

Motivación para descomponer un problema en partes

En la cultura popular, divide y vencerás hace referencia a una famosa metodología que implica resolver un problema difícil dividiéndolo en partes más simples tantas veces como sea necesario, hasta que la resolución de las partes se torna obvia. La solución del problema principal se construye con las soluciones encontradas.

Está comprobado que un problema complejo es más difícil de resolver que uno sencillo, y la complejidad de un problema global es la suma de las complejidades de cada una de sus partes.

La gestión de dependencias

Una de las principales características de la modularización es que, a pesar de que un módulo sea una entidad independiente pueda extender, o utilizar cierta funcionalidad almacenada en otro módulo distinto, y ahí aparece el problema de gestión de dependencias.

Una dependencia es una aplicación o librería requerida por otro programa para poder funcionar correctamente. Por esto, se dice que dicho programa depende de esa aplicación o librería. La gestión de dependencias consiste en la correcta administración de las dependencias de un programa.

Algunos ejemplos del éxito de la modularidad y la gestión de dependencias es el sistema operativo Ubuntu. Con Ubuntu y el gestor de paquetes Synaptic, podemos instalar fácilmente nuevos programas delegando en Synaptic la gestión de dependencias, la descarga de paquetes necesarios para la correcta ejecución del programa que queremos instalar. Otro ejemplo orientado a Java es el proyecto Eclipse, que proporciona mecanismos para ampliar y personalizar fácilmente la plataforma mediante el desarrollo de plugins.

2.3 Objetivos generales

e-Adventure es una herramienta para la creación de videojuegos que con el tiempo ha aumentado considerablemente la funcionalidad ofrecida. En el diseño original de e-Adventure no se tuvo en cuenta la extensión de la plataforma de manera genérica, de forma que aunque se pueden realizar cambios y aumentar la funcionalidad, estas extensiones tienen un límite. Durante varios años de desarrollo, se empiezan a registrar una serie de situaciones en las que no es posible, o resulta muy difícil, añadir ciertas nuevas funcionalidades en los juegos. Pero realmente la complejidad de este tipo de tareas no debería ser muy laboriosa si se hubiera tenido en cuenta en el diseño original de e-Adventure la posibilidad de extender siguiendo metodologías de diseño basadas en plugins.

Por este motivo, se empiezan a buscar soluciones que puedan solventar la problemática de ampliar la funcionalidad en e-Adventure, tomando también como referencia el trabajo relacionado realizado por otros desarrolladores de herramientas en general y de creación de videojuegos en particular ante problemas similares.

La solución correcta sería realizar un núcleo para la plataforma de creación de videojuegos con la funcionalidad básica necesaria para la creación de videojuegos simples pero que estuviera preparada para ampliarse fácilmente mediante extensiones, de forma que cualquier nueva funcionalidad que se quisiera introducir en la herramienta fuese modular, independiente del núcleo y de fácil inclusión en éste. Además, de esta forma también se facilitaría la creación de plugins o extensiones a programadores externos al equipo de e-Adventure, ya que conociendo la funcionalidad disponible del núcleo, cualquier persona podría ser capaz de programar su propia extensión sin necesidad de conocer todo el código en profundidad.

Dado que el proyecto e-Adventure está implementado utilizando tecnología Java nos centramos en este lenguaje a la hora de investigar alternativas de modularización y plugins.

Para que un sistema en Java gestione plugins de forma básica se deben realizar una serie de pasos:

1. Definir la forma en que los plugins serán agregados al sistema.
2. Crear una interfaz que los plugins deberán implementar.
3. Encontrar la forma de localizar y cargar los plugins (a poder ser en tiempo de ejecución).

En la actualidad existen a nuestra disposición frameworks que realizan estas tareas limitando nuestro trabajo a la adaptación de nuestra plataforma al framework y a la creación de los plugins.

Algunos ejemplos de estos frameworks son OSGI, Java Plugin Framework o Jin-plugin, que estudiaremos más a fondo en el siguiente capítulo.

2.3.1 Objetivos principales

En base a la decisión elegida de creación de un sistema (o framework) de extensiones, debido a los problemas anteriormente citados, surgen principalmente los siguientes objetivos a realizar.

Objetivo 1

Producción de un sistema de extensiones para la herramienta de desarrollo e-Adventure que permita implementar funcionalidades extra que puedan ser incorporadas de manera independiente al núcleo principal. Estas extensiones se tienen que poder añadir de forma sencilla y por cualquier persona que lo desee, tanto de dentro como de fuera del equipo de desarrollo de e-Adventure.

Por esto, e-Adventure se convertiría en una herramienta cuya funcionalidad se ajustara a las necesidades de cada individuo, evitando la creación de una herramienta pesada y difícil de manejar.

Además, permitirá la extensión de una plataforma de manera que los usuarios de e-Adventure puedan utilizar cada uno de los nuevos complementos sin necesidad de conocimientos técnicos.

Asimismo, como se trata de una herramienta de código abierto, todo el mundo podrá tener acceso a cualquier plugin desarrollado por cualquier programador, facilitándose así la creación de una comunidad de usuarios y desarrolladores de e-Adventure.

Objetivo 2

Desarrollo de plugins de ejemplo para probar la funcionalidad del framework. Se desarrollarán varios plugins sobre juegos educativos que nos

permitan ver la plataforma desde la perspectiva de los futuros programadores de plugins.

Con esto pretendemos conocer más a fondo la estructura e implementación de la nueva versión de e-Adventure y descubrir las ventajas que ofrece el framework de plugins e intentar solventar sus limitaciones.

Capítulo III
Estado del arte

Tabla de contenidos

3. Estado del arte

En este apartado se va a proceder a explicar cómo el estado del arte actual respecto a las áreas que van a ser tratadas en nuestro proyecto, como es el caso de las plataformas de creación de videojuegos y las plataformas para la gestión de plugins. Por lo tanto, el capítulo se divide en dos subsecciones, en las que se va a hablar de cada uno de los temas mencionados.

La primera subsección va a describir el panorama actual en plataformas para creación de videojuegos, es decir, aquellas herramientas que facilitan el desarrollo de juegos. Se establece una introducción sobre este tipo de aplicaciones, para seguidamente detallar ejemplos concretos de plataformas. Entre los ejemplos que se van a comentar, destacar la herramienta **e-Adventure**, aplicación elegida como pilar de nuestro proyecto para desarrollar todo el análisis y diseño de nuestro sistema de plugins. Se trata de una aplicación de la UCM, que facilita la creación de juegos educativos tipo 'point&click'.

En la siguiente subsección, se va a tratar el tema de las plataformas para la gestión de plugins. Se va a realizar una detallada descripción de distintas herramientas de este tipo, explicando sus características, funcionamiento y usos. Esta área influye directamente a nuestro proyecto, puesto que se elegirá de entre las plataformas estudiadas en esta sección, una de ellas para la elaboración de nuestro sistema de plugins. Por lo tanto, es muy importante establecer un análisis adecuado de estas herramientas, para poder posteriormente escoger aquella más adecuada a las necesidades y requisitos de nuestro sistema.

3.1 Plataformas de creación de videojuegos

La creación de videojuegos es una actividad multidisciplinar que involucra a profesionales de distintas ramas dentro de la informática (programadores, analistas, desarrolladores, etc.) y fuera de ella (técnicos de sonido, diseñadores, grafistas, etc.).

Para crear un videojuego se han de seguir ciertas fases, a saber:

- Concepción de la idea del videojuego
- Diseño
- Planificación
- Producción
- Pruebas
- Mantenimiento

Una plataforma de desarrollo de videojuegos es una aplicación de software especializado que ayuda o facilita la realización de un juego de ordenador. Algunas de las tareas a cargo de estas herramientas incluyen la conversión de modelos 3D, texturas, etc., en formatos requeridos por el juego. Las herramientas de juego pueden o no ser liberadas tras el lanzamiento del mismo. A día de hoy, se suelen incluir al menos editores de nivel para el juego.

En los primeros tiempos de la industria del videojuego no existían herramientas para el desarrollo de los mismos, si bien es cierto, que esto no fue un problema para el tipo de juegos que se realizaban en aquella época. Juegos como el Pac-Man disponían de niveles generados con un editor, pero estos eran implementados directamente en el código del juego.

Las imágenes de los caracteres también se implementaban directamente, siendo dibujadas frame a frame por los diferentes comandos del código. Según fue avanzando el nivel de la tecnología, se hizo cada vez más habitual el uso de sprites (un "sprite" es cualquier imagen o animación bidimensional) y empezaron a emerger algunas herramientas de ayuda al desarrollo.

A continuación, se citan algunas razones que favorecen el uso de plataformas que ayudan al desarrollo de videojuegos:

- Disminución del grado de dificultad técnica en el desarrollo.
- Menor tiempo de trabajo en la realización de un juego.
- Acercamiento a un público más amplio y menos especializado: Reducir la dificultad de la implementación de videojuegos implica que una mayor cantidad de gente y con un menor grado de especialización puedan hacerlos. El objetivo es que personas sin experiencia alguna en el entorno de la programación puedan desarrollar sus propios videojuegos.
- Interés en desarrollar videojuegos propios con intención de venta, uso personal o libre distribución: En referencia al apartado anterior, una persona que tendría imposible sin una herramienta de desarrollo el generar un juego podrá hacerlo ahora con mucho menor esfuerzo, lo que generará un mayor abanico de motivación en los videojuegos, no solamente comercial. Un ejemplo de esto podría ser un fin educativo concreto de un profesor para con sus alumnos.

Se pueden distinguir diferentes tipos de plataformas:

A. Motores: Se entiende por motor a un sistema software diseñado para la creación y desarrollo de videojuegos. El principal ejemplo de motor es OGRE (Object-orientedGraphicsRenderingEngine)[12]. OGRE es un motor interpretado de gráficos de código abierto que permite, entre otras funcionalidades:

   - Interfaces Orientadas a Objetos (OO) fáciles de usar, diseñadas para minimizar el esfuerzo requerido para interpretar escenas 3D y poder ser independientes de la implementación 3D.
   - Framework de ejemplo que permite obtener una aplicación que se ejecute de manera rápida y simple.
   - Diseño totalmente orientado a objetos para permitir extender la funcionalidad del motor con plugins y subclases de una manera sencilla.

B. Frameworks: Se entiende por framework a un marco de aplicación o conjunto de bibliotecas orientadas a la reutilización a muy gran escala de componentes software para el desarrollo rápido de aplicaciones. Como ejemplo de framework podemos citar Microsoft XNA[13]. XNA está compuesto por un conjunto de librerías y herramientas que permiten desarrollar videojuegos de forma fácil sobre las plataformas Windows, Xbox 360 y Zune. XNA está compuesta principalmente por:

   - XNA Framework se basa en la implementación nativa de .NET Compact Framework para el desarrollo en Xbox 360 y Zune, para el desarrollo bajo entorno Windows se apoya sobre el .NET Framework. Incluye un amplio conjunto de bibliotecas de clases, específicos para el desarrollo de juegos, por ejemplo para el manejo de dispositivos de entrada, tratamiento de sonidos y vídeos, carga de modelos y texturas, uso de ficheros de forma transparente a la plataforma en la que se ejecute, desarrollo de juegos online, etc.
   - XNA Game Studio se integra perfectamente con Visual Studio para facilitar el desarrollo de videojuegos usando XNA Framework, para ello proporciona distintas herramientas, instala plantillas especiales en Visual Studio, facilitando la creación de nuevos proyectos XNA, la carga de contenido como Texturas, Efectos, Videos, Sonidos y facilitando la conversión de proyectos entre las distintas plataformas.

C. Herramientas de autoría sin programación: Las herramientas de autor, también llamadas lenguajes de autor o software de autor, son un tipo de software compuesto por formatos o plantillas para diseñar material didáctico con distinto grado de interactividad que permite elaborar

12 Página web oficial de OGRE http://www.ogre3d.org
13 Página web del desarrollo de Microsoft XNA http://create.msdn.com/en-US/

archivos de tipo gráfico, audio, vídeo, etc. Se trata de aplicaciones informáticas que permiten realizar un proceso de enseñanza-aprendizaje multimedia. Un ejemplo de herramienta de autoría sería GameSalad, que permite la creación de juegos para MAC y del que se hablará en los próximos apartados.

D. Herramientas de autoría con programación: Una herramienta de autor con programación no es más que una herramienta que consta de plantillas para elaborar diferentes tipos de archivos pero que, al contrario que las herramientas de autor sin programación, sí es requiere en mayor o menor grado ciertas tareas de programación. En este caso, podemos citar Unity, del que también se hablará un poco más adelante, como ejemplo de este tipo de herramientas

3.1.1 Ejemplos de plataformas de creación de videojuegos

A continuación vamos a comentar algunas de las plataformas más usadas actualmente. Para la concepción de esta lista nos hemos basado en dos sistemas de categorización para añadir el grado de relevancia de cada plataforma:

- El buscador de Google (http://www.google.com): Siendo el principal buscador a nivel global en estas fechas nos parece indispensable usarlo como herramienta para el chequeo de la importancia de cada plataforma.
- Delicious (http://www.delicious.com): Se trata de un servicio de gestión de marcadores sociales en web. Permite agregar los marcadores que clásicamente se guardaban en los navegadores y categorizarlos con un sistema de etiquetado denominado folcsonomías o tags.

a) e-Adventure[14]

Empezamos comprobando el grado de relevancia de la herramienta desarrollada por el equipo de la Universidad en el que trabajamos. La plataforma e-Adventure es un proyecto de investigación que aspira a facilitar la integración de juegos educativos y simulaciones basadas en juegos en procesos educativos en general y Entornos Virtuales de Aprendizaje (VLE) en particular.

Número aproximado de resultados en Google: 29.100
Número de resultados en Delicious: 45

Características principales:

- Soporte para todos los rasgos comunes de aventuras gráficas point&click y de ficción interactiva.

14 Plataforma de creación de videojuegos E-Adventure http://e-adventure.e-ucm.es

- Multiplataforma: motores para PC, Macintosh OS X y Android (en estado "beta").
- Juegos empaquetados con metadatos estándares (IEEE LearningObjectMetadata, LOM-ES).
- Integración con LMS a través de la implementación de diversos estándares educativos (SCORM1.2 y 2004 e IMS Content Packaging), exportación especial para la integración con WebCT 4.0 y soporte para IMS LearningDesign.
- Opensource (código abierto).
- Escrito en Java.
- Puede ser desplegado como una aplicación independiente, o como un applet para la educación online.
- No soporta plugins.

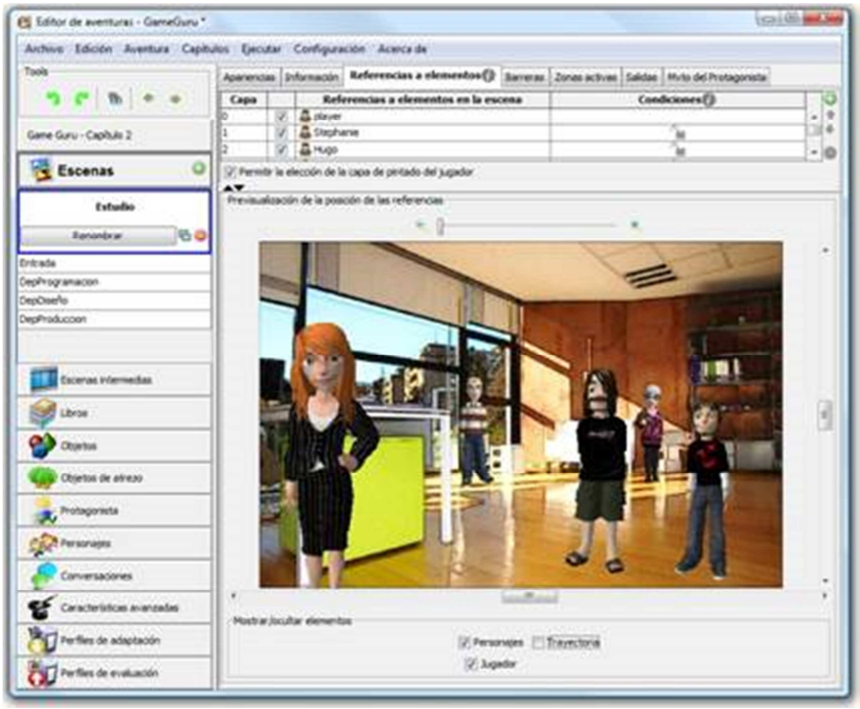

Figura 7. Imagen de la plataforma e-Adventure

b) Adventure maker[15]

Herramienta de tipo "point-and-click" que permite crear videojuegos para Windows, PSP, iPhone e iPod touch.

Número aproximado de resultados en Google: 29.300 resultados
Número de resultados en Delicious: 110

Características principales:

- Para la creación de juegos de aventura, visitas virtuales interactivas, software educativo, etc.

15 Plataforma de creación de videojuegos Adventure Maker http://www.adventuremaker.com

- Tipo point and click.
- 3 dimensiones.
- Versión de pago y gratuita, que contiene todas las características principales.
- Multiplataforma: para Windows, PSP, iPhone e iPod Touch.
- Incluye herramientas de composición de música y de dibujo.
- Dispone de plugins (extensiones) tipo OCX de ActiveX para la versión de Windows.

Los plugins proporcionados por Adventure Maker solo es posible incorporarlos para Windows. En la creación de aplicaciones para PSP, iPhone o iPod Touch, se está considerando para futuras versiones, pero actualmente no está disponible. Además, ofrecen la opción de poder desarrollar plugins, facilitando un tutorial y documentación para realizarlos. Los plugins pueden ser enviados para su incorporación en la página web de la plataforma y todos los usuarios puedan hacer uso de ellos.

Figura 8. Imagen de la plataforma Adventure Maker

Los plugins se pueden crear de dos formas distintas[16]. La primera, la más sencilla, se hace desde la propia aplicación, a través de la barra de menús, existe una opción que es la de crear nuevo plugin. Muestra una ventana de propiedades, donde se configuran las diferentes características que deseemos extender con el plugin. Este tipo de plugins solamente contienen código nativo VBScript del motor de Adventure Maker. La otra forma de creación hace uso de la personalización de componentes ActiveX, ejecutables que pueden lanzarse dentro de otros ejecutables. Para su instalación, simplemente es necesario copiarlos a la subcarpeta 'Plugins' dentro de Adventure Maker y posteriormente, habilitarlos en la ventana de propiedades del proyecto.

Algunos de los plugins más interesantes con los que cuenta esta herramienta son:

16 Sistema de plugins de Adventure Maker
http://www.adventuremaker.com/help/plugins_overview.htm

- Plugin para lanzar eventos aleatorios
- Plugin de botones de inventario
- KeyGuard: Consiste en un plugin que permite el uso del teclado y del botón derecho del ratón para lanzar acciones, proporciona eventos para controlar si el ratón está dentro o fuera de una determinada área, etc.

Figura 9. Imagen de la plataforma Adventure Maker

- SUBMaster: Permite incorporar fácilmente subtítulos en distintos idiomas a los proyectos
- EZTalk: Facilita toda la gestión de diálogos y conversaciones con múltiples opciones, ofreciendo la posibilidad de lanzar archivos de audio
- ChangesMessages Font: Nos permite personalizar la fuente de los mensajes
- ThirdPersonplugin: Presenta la opción de crear juegos en tercera persona

c) JClic[17]

JClic es un entorno para la creación, realización y evaluación de actividades educativas multimedia, desarrollado en la plataforma Java. Es una aplicación de software libre basada en estándares abiertos que funciona en diversos entornos operativos: Linux, Mac OS X, Windows y Solaris.

Número aproximado de resultados en Google: 182.000 resultados
Número de resultados en Delicious: 1504

Características principales:

- Herramienta para la creación de aplicaciones didácticas multimedia.
- Hacer posible el uso de aplicaciones educativas multimedia "en línea", directamente desde Internet.

17 Plataforma de creación de videojuegos JClic http://clic.xtec.cat/es/jclic/index.htm

- Continuación del proyecto Clic 3.0 y total compatibilidad con sus aplicaciones existentes.

Figura 10. Imagen de la plataforma JClic

- Uso en diversas plataformas y sistemas operativos, como Windows, Linux, Solaris o Mac OS X.
- Utiliza un formato estándar y abierto para el almacenaje de los datos, con el fin de hacerlas transparentes a otras aplicaciones y facilitar su integración en bases de datos de recursos.
- Ampliar el ámbito de cooperación e intercambio de materiales entre escuelas y educadores de diferentes países y culturas, facilitando la traducción y adaptación tanto del programa como de las actividades creadas.
- La herramienta de programación escogida ha sido Java, y el formato para almacenar los datos de las actividades es XML.
- Licencia Creative Commons (código abierto).
- Desarrollado por la Generalitat de Cataluña.
- No permite extender la funcionalidad con plugins.

Figura 11. Imagen de la plataforma JClic

d) RPG Toolkit[18]

RPG Toolkit es una aplicación software de código gratuito y libre que permite crear juegos de rol para PC. La página web actúa no sólo como host del software sino también como base a la comunidad de usuarios y desarrolladores.

Número aproximado de resultados en Google: 8.310
Número de resultados en Delicious: 57

Características principales:

- Para la creación de juegos de rol.
- Solamente disponible en PC.
- 2D.
- No permite extender la funcionalidad con plugins.

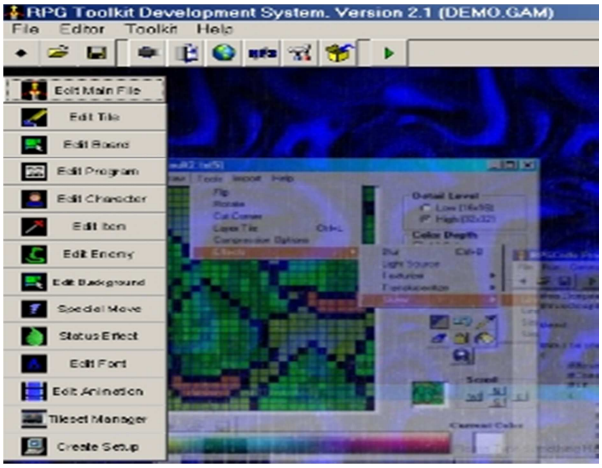

Figura 12. Imagen de la plataforma RPG Toolkit

Figura 13. Imagen de la plataforma RPG Toolkit

18 Plataforma de creación de videojuegos RPG Toolkit http://www.toolkitzone.com/index.php

e) Game editor[19]

Game editor es una herramienta de diseño de videojuegos con licencia open source. Permite el diseño e implementación de juegos 2D para ordenadores personales y dispositivos móviles.

Número aproximado de resultados en Google: 324.000
Número de resultados en Delicious: 123

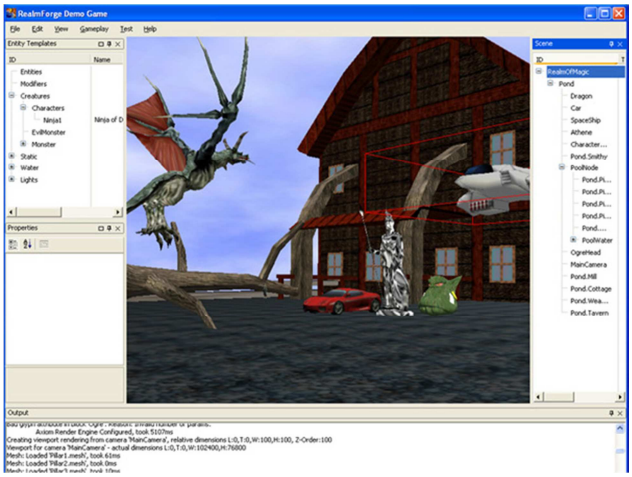

Figura 14. Imagen de la plataforma Game Editor

Características principales:

- Open source bajo licencia GPL v3. Pero también hay versión de pago si no interesa seguir con la licencia para vender los juegos creados.
- Multiplataforma. Una vez el juego este creado se pueden generar ejecutables para windows, Pocket PC/Windows Mobile, Handheld PC, Windows Mobile-basedSmartphones, GP2X y Linux.
- Poca programación necesaria.

Figura 15. Imagen de la plataforma Game Editor

19 Plataforma de creación de videojuegos Game Editor http://game-editor.com

- Soporta prácticamentetodos los tipos de archivos de audio, imagen, sonido,etc.
- Puedes vender los juegos en su página web (su propio market).
- Juegos 2d.
- Multiplataforma: para PC y Mac, iPad, iPhone, iPod.
- No permite extender la funcionalidad con plugins.

f) Adventure game studio[20]

Aplicación que permite crear juegos de aventura del tipo "point-and-click". Consiste en un editor fácil de usar para crear los juegos y un motor para jugar a ellos. La interfaz de juego es totalmente personalizable.

Número aproximado de resultados en Google: 38.800
Número de resultados en Delicious: 31

Características principales:

- Plataforma para crear videojuegos del estilo "point-and- click".
- Código libre.
- Multiplataforma: Windows, Linux (sólo ejecución), Mac OS X (sólo ejecución).
- Plugins: funcionan en las tres plataformas pero en Linux y Mac dan bastantes problemas. Soporta dos tipos:
    - API plugins VC++ para escribir plugins que añaden funcionalidades extra a los juegos. Están escritos mayoritariamente en C++ y Delphi. Se implementan como archivos DLL de Windows, cuyo nombre debe comenzar por "AGS". Además, se necesita crear el plugin como un proyecto estándar DLL de Windows. Cuando la aplicación se inicie, leerá todos los ficheros AGS*.DLL en el directorio del editor, y los añadirá a la lista de plugins. Un plugin puede ser lanzado de dos maneras; design-time, usado cuando el plugin es cargado desde el editor AGS, y run-time, modo empleado cuando el juego se está ejecutando.
    - .NET plugins, utilizados para mejorar el editor AGS. Pueden ser escritos en cualquier lenguaje .NET, incluyendo C#, VB.NET, Visual C++ .NET, entre otros[21].

Ejemplos de plugins desarrollados para AGS:

- Character 3D: Añade la funcionalidad de poder utilizar modelos de personajes 3D.
- CharacterControl System: Permite un mayor control respecto a los personajes. Informa sobre qué hacer en un momento determinado, cómo moverlos, quién se encuentra en otras habitaciones, etc...

20 Plataforma de creación de videojuegos Adventure Game Studio
http://www.adventuregamestudio.co.uk
21 Sistema de plugins de Adventure Game Studio
http://www.adventuregamestudio.co.uk/accomplug.htm

- Collision Detector: Aporta funciones de detección de colisiones para los sprites.
- Fire: Plugin que facilita la creación de diferentes efectos con fuego.
- Snow\Rain plugin: Proporciona efectos con nieve y lluvia.
- TCP/IP-Pluginfor AGS: Plugin de gestión de la red para el editor AGS. Comenta que su uso es bastante complicado.

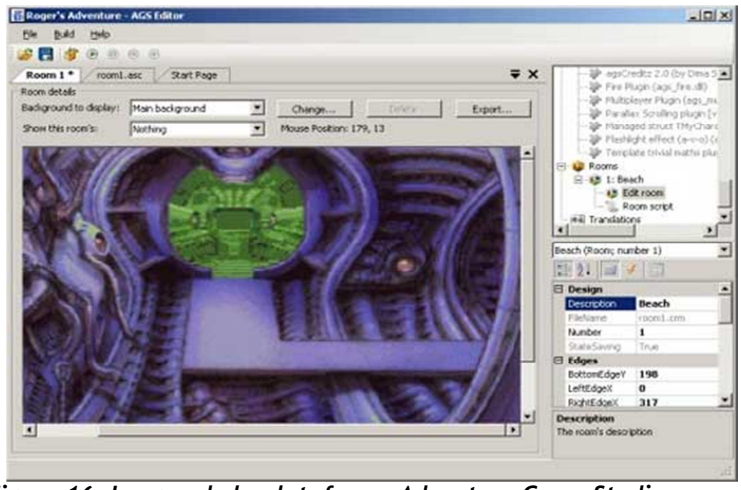

Figura 16. Imagen de la plataforma Adventure Game Studio

g) Basic4gl[22]

Basic4GL es un lenguaje de programación gratuito para ordenadores Windows con apoyo de la librería gráfica OpenGL. Está basado en BASIC. Fue diseñado para ser un lenguaje fácil de aprender y usar sin necesidad de la instalación de todos los compiladores asociados a este tipo de programas.

Número aproximado de resultados en Google: 111.000
Número de resultados en Delicious: 13

Características principales:

- Plataforma para Windows con soporte incorporado para la biblioteca de gráficos OpenGL.
- Programas en 3D, juegos, demos, etc.
- Soporta plugins que permiten mejorar el lenguaje con nuevos comandos y funciones. Hay una serie de plugins existentes creadas por los miembros de la comunidad Basic4GL. Se pueden crear usando C++ mediante el SDK de Basic4GL. Para poder utilizarlos, los ficheros correspondientes del plugin se deben situar en la carpeta Basic4GL y después ejecutar la aplicación y hacer click en File->Plug-in Libraries.

Algunos plugins para esta herramienta:

22 Plataforma de creación de videojuegos Basic4gl http://www.basic4gl.net/

- ActiveX Decoder: Se trata de un plugin para decodificar ficheros de audio (*.wav, *.mp3, *.mid), y de vídeo (*.avi, *.mpg, *.wmv *.mp4).
- ColDet Collision Detection: Detector de colisiones.
- GLU OpenGLUtility Library: Añade funcionalidad de la librería gráfica OpenGL.
- Newton Game Dynamics Plugin: Proporciona un motor de física para poder emplear simulaciones dinámicas en los juegos.
- TinyConsole: Añade una consola de texto y comandos necesarios para su uso.
- Tiny Font: Añade soporte para la gestión de fuentes.

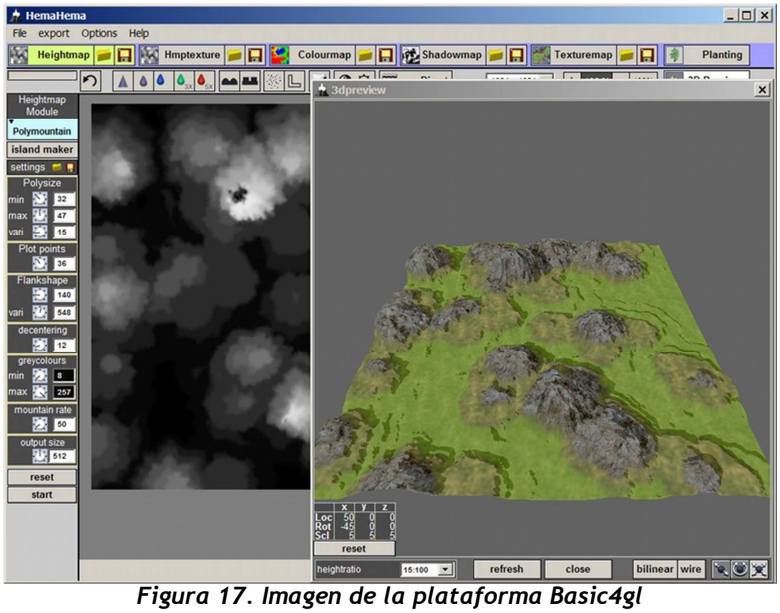

Figura 17. Imagen de la plataforma Basic4gl

h) Game Maker[23]

GameMaker permite hacer videojuegos sin la necesidad de escribir una sola línea de código. Hace uso del sistema "arrastrar y soltar" con el que se pueden crear juegos con un acabado profesional en un periodo corto de tiempo. Se pueden insertar fondos, gráficos animados, música e incluso juegos en 3D.

Número aproximado de resultados en Google: 1.650.000
Número de resultados en Delicious: 1633

Características principales:

- No es necesario aprender un lenguaje de programación pero para usuarios experimentados contiene un lenguaje de programación llamado "GML" o "GameMakerLanguage" basado en built-in scripting.

23 Plataforma de creación de videojuegos Game Maker http://www.gamemaker.nl

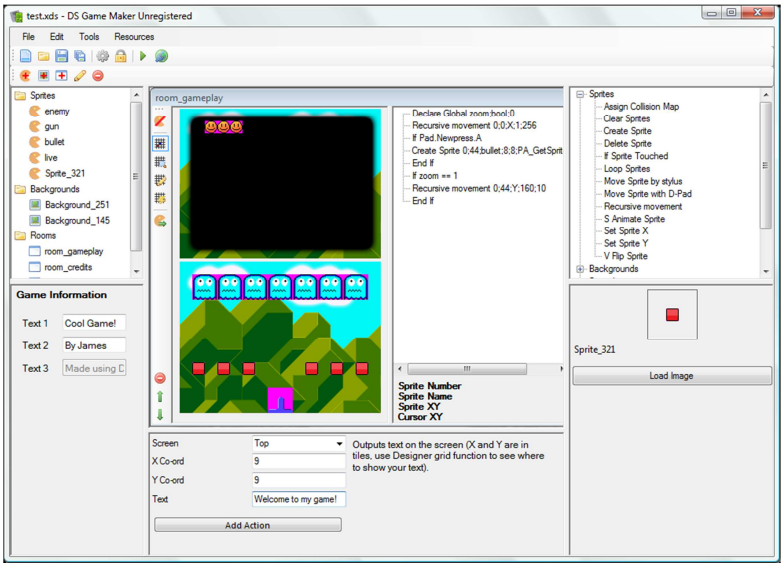

Figura 18. Imagen de la plataforma Game Maker

- Los juegos se pueden distribuir sin licencia o con licencia como ejecutables (.exe) o código fuente (GMK. GM6. GMD. GMF).
- Versión gratuita y de pago con mayor funcionalidad (capacidad de incorporar archivos DLL, crear juegos multijugador, Direct3D para crear shooters en primera persona, etc.).
- No permite extender la funcionalidad con plugins.

i) Unity[24]

Unity es una herramienta de desarrollo de videojuegos que ha sido diseñado para centrarse en la creación de juegos. Se trata de un entorno de desarrollo que permite que el usuario se concentre simplemente en hacer el juego. Además se pueden desarrollar videojuegos tanto para videoconsolas, dispositivos móviles como Internet.

Número aproximado de resultados en Google: 828.000
Número de resultados en Delicious: 4757

Características principales:

- Es simple de usar.
- Se ha puesto especial esfuerzo en herramientas de creación de mundos para poder montar y ajustar niveles según vayan surgiendo las ideas.
- Testear una idea no requiere más que unos clicks lo que implica poder centrarse en el videojuego y poder pulirlo fácilmente.
- Sencillo a la hora de cambiar la plataforma por lo que se puede desarrollar en una sola fuente para poder importarlo después en otra (PC/Mac, móviles, en internet, consolas, etc.)
- Es potente.

24 Plataforma de creación de videojuegos Unity3d http://unity3d.com

- Para la última versión (Unity 3) se ha mejorado el canal de renderizado para conseguir un incremento de hasta el 50% en el rendimiento.
- Implementa "Beastlightmapping" y oclusión Umbra para que los videojuegos vayan más rápidos en cualquier dispositivo.
- Se han incluido herramientas de audio y de edición "entre escenas" para diseñar un mejor ambiente en el videojuego.
- Dispone de un sistema de plugins para incrustar los juegos en los principales navegadores web. Está basado en:
    - Escribir el plugin deseado en un lenguaje basado en C y compilarlo.
    - Crear un script C# que llame a la función del plugin para realizar la funcionalidad deseada.

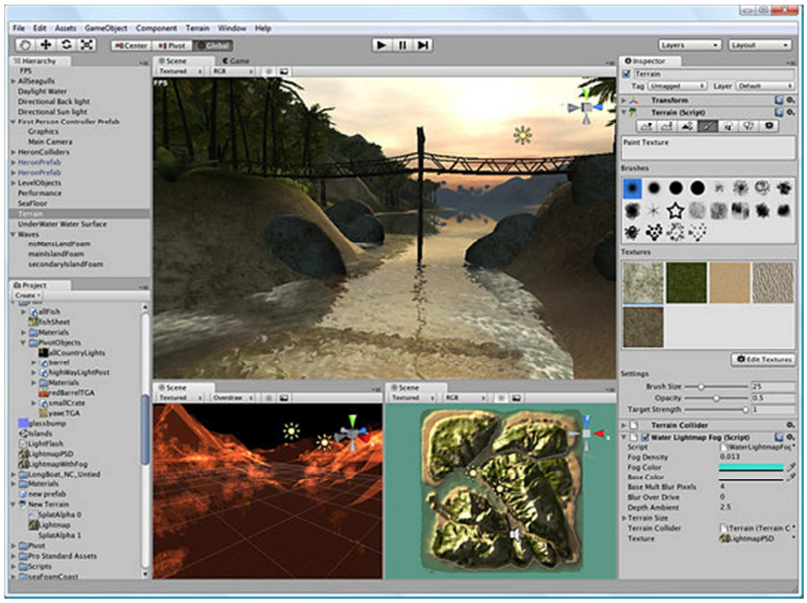

Figura 19. Imagen de la plataforma Unity

Ejemplos de extensiones que ofrece:

- TerrainToolkit: Integra un conjunto de herramientas para el editor Unity, las cuales facilitan la creación de terrenos realistas para los juegos.
- Explosion Framework: Aporta efectos de explosión, pudiendo configurar su color, tamaño, duración, y la aparición de partículas como humo, ondas, etc...
- LocomotionSystem: Hace que los personajes de los juegos caminen y corran por cualquier tipo de terreno irregular, de forma que parezca natural y correcto.

j) Game Salad[25]

Herramienta de creación de videojuegos para no-programadores. Se trata de una herramienta, diseñada para personas con cualquier nivel de

25 Plataforma de creación de videojuegos Game Salad http://gamesalad.com

conocimientos técnicos, en el que no es necesaria incluir ni una línea de código. Se construyen videojuegos visualmente con una interfaz de "arrastrar y soltar" con un comportamiento del sistema muy robusto. Permite publicar juegos rápida y fácilmente en iPhone, iPod Touch e iPad.

Número aproximado de resultados en Google: 37.400
Número de resultados en Delicious: 143

Características principales:

- No requiere código.
- El juego se crea visualmente. Se comprueban los cambios introducidos en tiempo real.
- Arrastrar y soltar. Se puede importar cualquier archivo propio fácilmente simplemente arrastrando desde el fichero contenedor.
- Publicación en la tienda virtual iTunes AppStore.
- Se puede compartir el juego de manera online para un mejor testeo del mismo.
- Permite crear juegos innovadores integrando mecanismos de física real.
- Posibilidad de crear acciones complejas mediante el uso de la "librería de comportamiento".
- Uso de funciones matemáticas para implementar complejos comportamientos de Inteligencia Artificial.

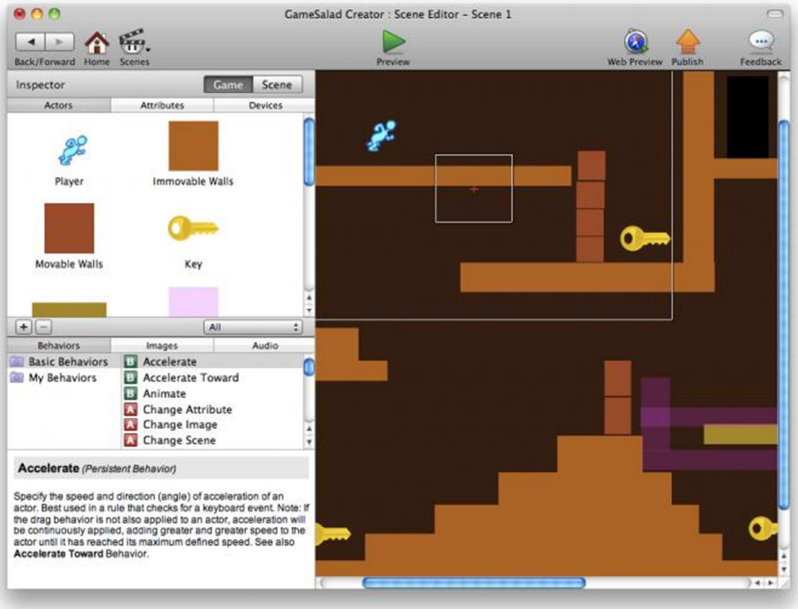

Figura 20. Imagen de la plataforma Game Salad

k) Alice[26]

Alice es un entorno de programación 3D que facilita la creación de animaciones para contar historias o cuentos, realizar juegos interactivos,

26 Plataforma de creación de videojuegos Alice http://www.alice.org

videos, etc. Se trata de una herramienta de aprendizaje hacia la computación. Usa gráficos 3D y una interfaz del tipo "arrastrar y soltar" para facilitar una primera experiencia.

Número aproximado de resultados en Google: 21.000
Número de resultados en Delicious: 57

Características principales:

- Herramienta diseñada para cualquier tipo de personas, desde expertos programadores a gente que no tenga conocimientos de programación.
- Interfaz sencilla e intuitiva.
- Diseñada como una primera toma de contacto con la programación orientada a objetos, permitiendo a los estudiantes a aprender tareas básicas de ingeniería informática.
- Opensource.
- Disponible para Windows, Linux y Mac.

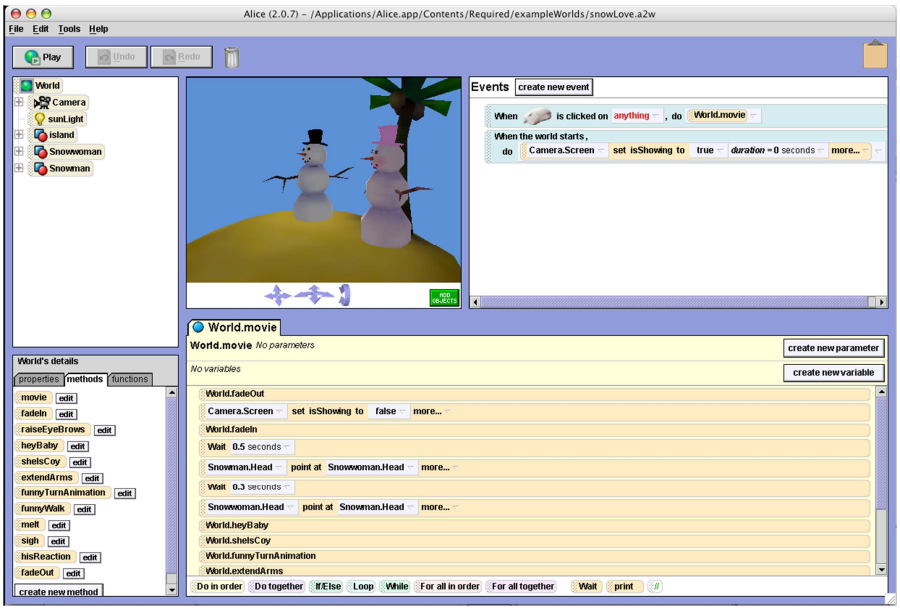

Figura 21. Imagen de la plataforma Alice

l) Raptivity[27]

Herramienta dividida en tres dominios según la utilidad deseada; eLearning, Presenter y WebExpert. Nos centramos en Raptivity eLearning por ser la herramienta con la funcionalidad acorde al presente trabajo. Se trata de una herramienta que permite crear "interacciones" de alta calidad que pueden ser fácilmente embebidos (empotrados) en cualquier herramienta de e-learning.

Número aproximado de resultados en Google: 37.400
Número de resultados en Delicious: 198

27 Plataforma de creación de videojuegos Raptivity http://www.raptivity.com

Características principales:

- Para Windows.
- Disponen de diferentes tipos de recursos: Educación, Salud y Alta tecnología.
- Contiene una librería con más de 200 “interacciones” de aprendizaje.
- Las interacciones creadas se pueden importar como un simple fichero Flash.
- Permite enviar información relativa a la interacción del jugador al servidor en el que está alojado siguiendo estándares educativos (en concreto, SCORM/AICC).
- Licencia privada, es decir, no se trata de una herramienta de software libre.

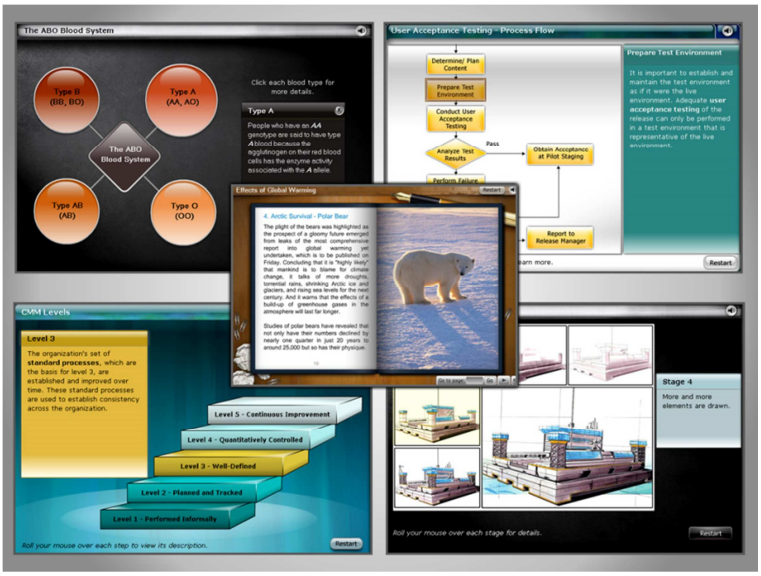

Figura 22. Imagen de la plataforma Raptivity

m) Scratch[28]

Es un lenguaje de programación que facilita crear historias interactivas, animaciones, juegos, música y arte. Además, permite compartir las creaciones con otras personas a través de Internet. La idea principal es aprender importantes nociones matemáticas y computacionales al mismo tiempo que se aprende a pensar creativamente, a razonar sistemáticamente y a trabajar colaborativamente.

Número aproximado de resultados en Google: 235.000
Número de resultados en Delicious: 482

Características principales:

- Herramienta multiplataforma.

28 Plataforma de creación de videojuegos Scratch http://scratch.mit.edu

- Scratch lo desarrolla el Lifelong Kindergarten Group en el Laboratorio de Medios del MIT (Instituto Tecnológico de Massachusetts).
- Licencia de uso libre Creative Commons "Attribution - Share Alike".
- Principalmente orientado a niños entre 6 y 16 años. Intenta ser lo más sencillo posible, limitando las funciones que se pueden realizar.
- ScratchMods. Se han desarrollado diferentes variaciones que incluyen alguna funcionalidad extra dependiendo de cada una de ellas. Algunos ejemplos son BYOB, Panther, Streak o Slash.

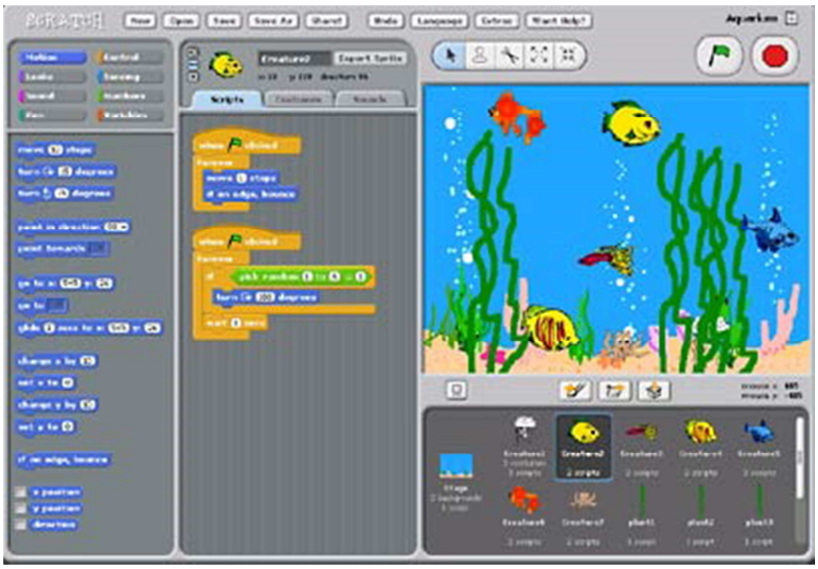

Figura 23. Imagen de la plataforma Scratch

Cuadro resumen de las herramientas

HERRAMIENTA	CARACTERÍSTICAS PRINCIPALES
e-Adventure	1. Multiplataforma (PC, MAC, etc.). 2. Juegos empaquetados con metadatos estándares (IEEE LearningObjectMetadata, LOM-ES). 3. Opensource.
AdventureMaker	1. Multiplataforma: para Windows, PSP, iPhone e iPod Touch. 2. Herramientas de composición de música y de dibujo. 3. Plugins tipo OCX de ActiveX para la versión de Windows.
JClic	1. Multiplataforma: Windows, Linux, Solaris o Mac OS X. 2. Formato estándar y abierto con el fin de hacerlas transparentes a otras aplicaciones. 3. Licencia CreativeCommons (Opensource).
RPG Toolkit	1. Creación de juegos de rol. 2. 2 dimensiones.

GameEditor	1. Open source bajo licencia GPL v3. 2. Multiplataforma: para PC y Mac, iPad, iPhone, iPod. 3. Los juegos se pueden vender en su página web.
AdventureGame Studio	1. Open source. 2. Multiplataforma: Windows, Linux, Mac OS X. 3. Plugins.
Basic4gl	1. Para Windows con soporte incorporado para la biblioteca de gráficos OpenGL. 2. Soporta plugins.
GameMaker	1. No es necesario aprender un lenguaje de programación pero para usuarios experimentados contiene un lenguaje de programación llamado "GML". 2. Versión gratuita y de pago con mayor funcionalidad.
Unity	1. 3 Dimensiones. 2. Multiplataforma: PC/Mac, móviles, en internet, consolas, etc. 3. Orientado a programadores.
Game Salad	1. No requiere código. El juego se crea visualmente. 2. Publicación en la tienda virtual iTunes AppStore. 3. Sólo para MAC.
Alice	1. Herramienta diseñada para cualquier tipo de personas, desde expertos programadores a gente que no tenga conocimientos de programación. 2. Interfaz sencilla e intuitiva. 3. Open source. 4. Multiplataforma: Windows, Linux y Mac.
Raptivity	1. Para Windows. 2. Licencia privada (de pago). 3. Contiene una librería con más de 200 “interacciones” de aprendizaje que se pueden modificar.
Scratch	1. Multiplataforma: Windows 2000 o superior, Mac OS X 10.4 o posterior, Ubuntu Linux 9.04 - 10.04. 2. Licencia de uso libre Creative Commons "Attribution - Share Alike". 3. Orientado a niños.

3.1.2 Análisis detallado de las herramientas más importantes

A continuación se presenta un análisis de las herramientas que consideramos más importantes de las previamente comentadas. El factor principal de la elección de cada una de ellas se ha basado en el mayor número de usuarios que utilizan cada una de ellas. También se ha tenido en cuenta que una herramienta desarrollada por una organización, universidad o gran empresa tiene una mayor repercusión que una herramienta desarrollada por una pequeña empresa.

JClic

Opiniones de la plataforma:

- Puntos fuertes:
  - Muy sencillo.
  - Se trata del desarrollo de juegos básicos que se van encadenando.
  - Posibilidad de importar como applet.
  - Herramienta para crear un instalador del proyecto.
  - Existe un módulo de JClic para poder exportar proyectos a moodle.
- Puntos débiles:
  - No existe la posibilidad de crear proyectos complejos.
  - Hay que basarse en el tipo de juegos que vienen con la herramienta.

Capturas de pantalla:

- Menú inicial

Se trata de la pantalla de inicio, donde podemos elegir el nombre del proyecto así como otras opciones informativas (Figura 24).

- Menú de mediateca

En el apartado de mediateca es donde se cargan los archivos externos que luego usaremos en el proyecto, esto es, imágenes, sonidos, videos, etc. (Figura 25).

- Menú de actividades

En este apartado es donde se realizan los juegos en sí.

Se puede elegir el tipo de actividad que queremos hacer entre todas las que nos ofrece JClic por defecto, o bien cargar alguna propia que el usuario haya creado por sí mismo (Figura 26).

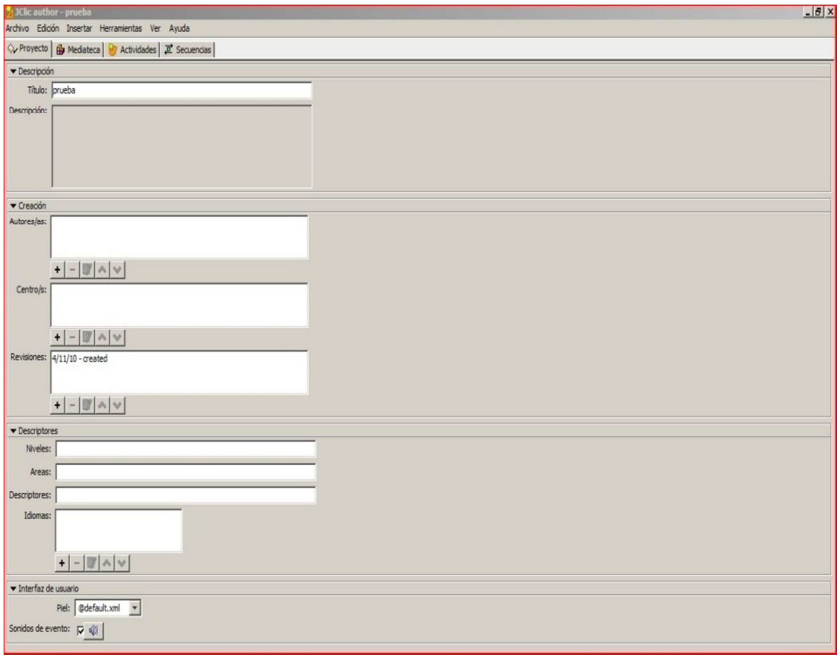

Figura 24. Menú principal JClic

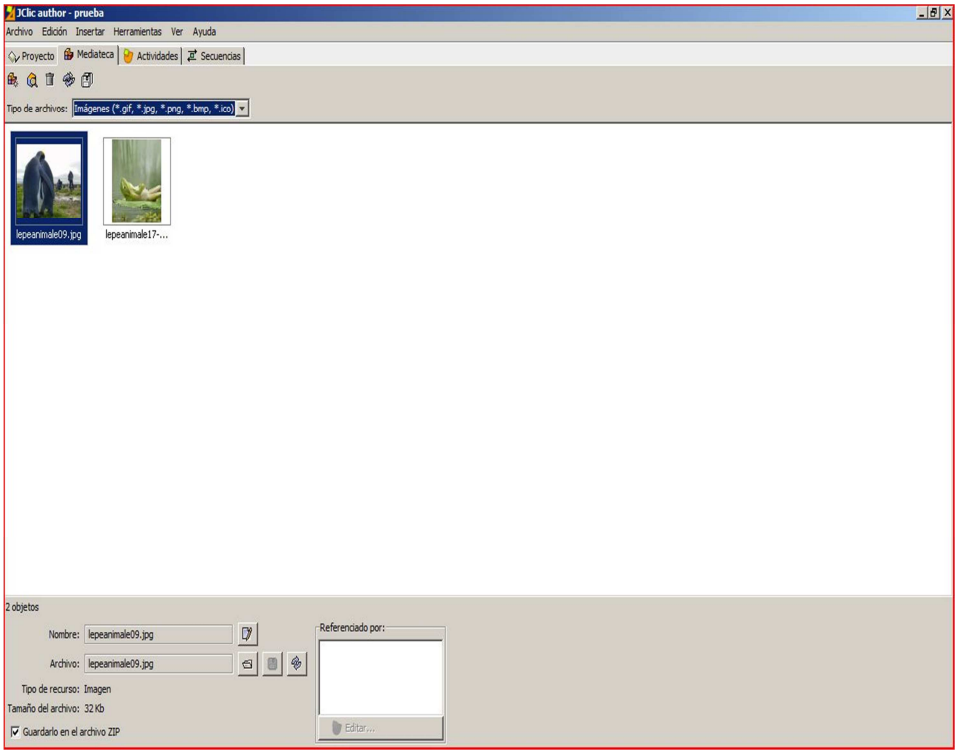

Figura 25. Menú de mediateca de JClic

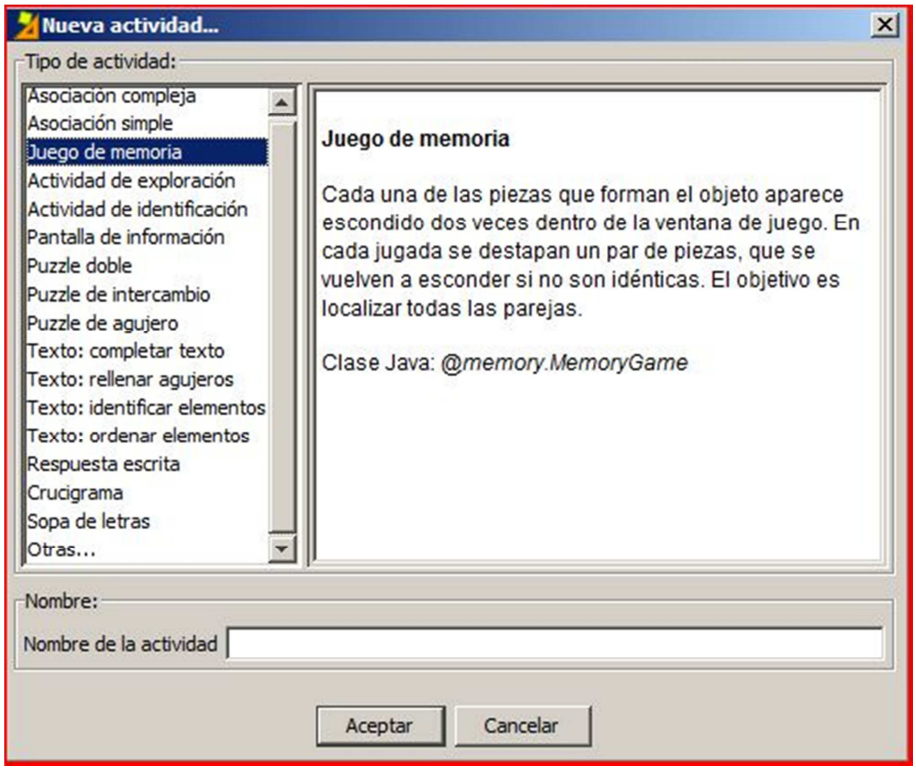

Figura 26. Pantalla de selección de actividades de JClic

El menú de actividades (Figura 27) es el que se muestra en la siguiente figura y, en él, podremos rellenar la actividad seleccionada según el juego concreto que queramos realizar.

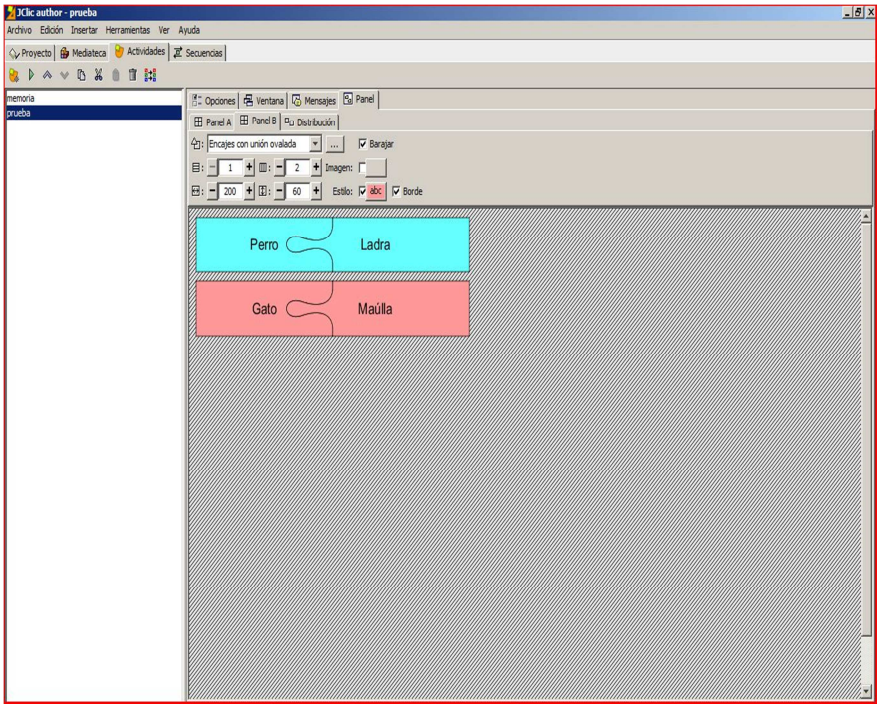

Figura 27. Menú de actividades de JClic

- Menú de secuencias

El menú de secuencias (Figura 28) nos permite crear el orden en el que queremos que el juego formado por las diferentes actividades, se ejecute.

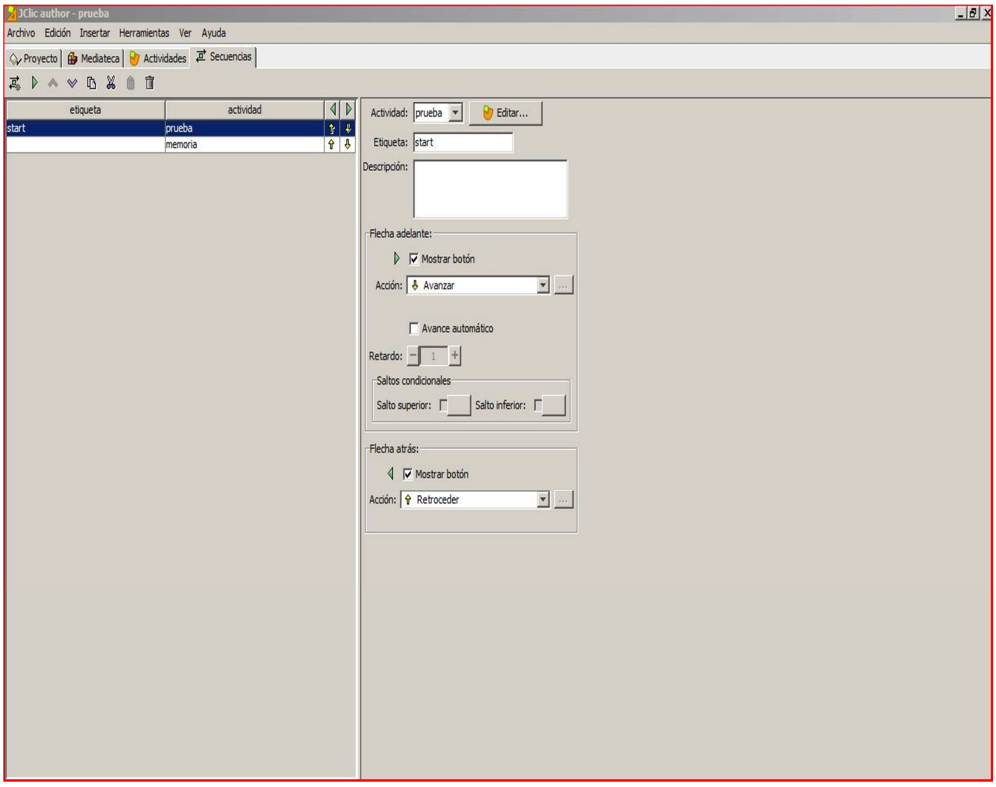

Figura 28. Menú de secuencias de JClic

- Pantallas de juego

A continuación se observa (Figura 29) una muestra de una actividad en la que se han de relacionar dos pares de conceptos mediante el uso de flechas.

Como podemos observar en la figura (Figura 30), en la esquina inferior derecha sale el número de aciertos, intentos y el tiempo que llevamos realizando la actividad.

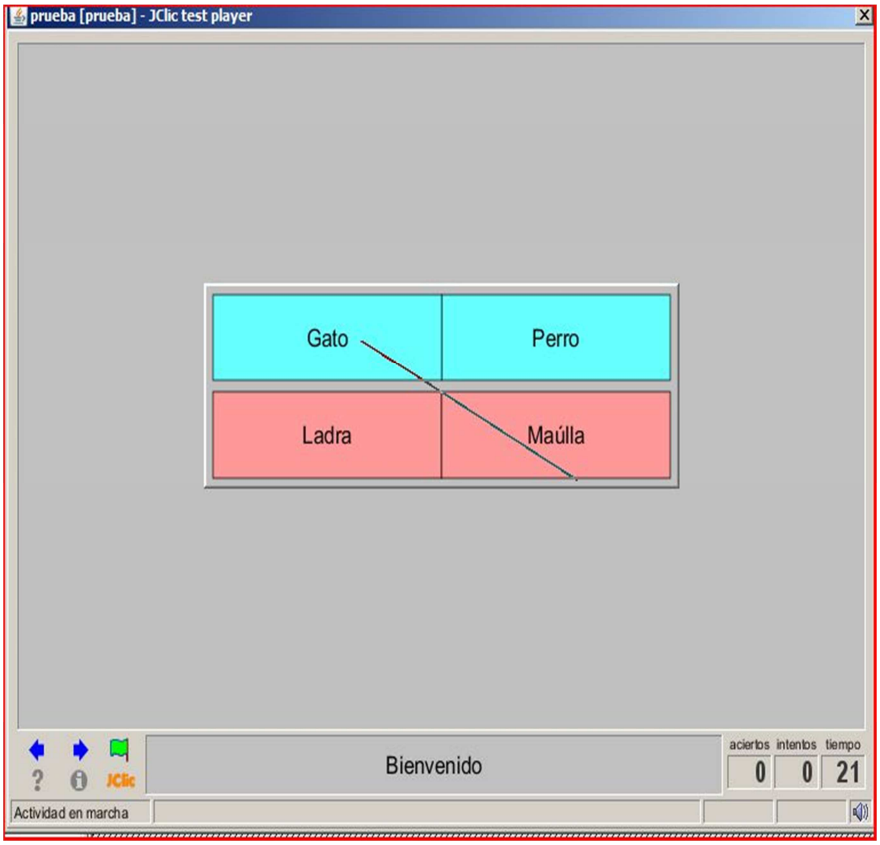

Figura 29. Pantalla de juego 1

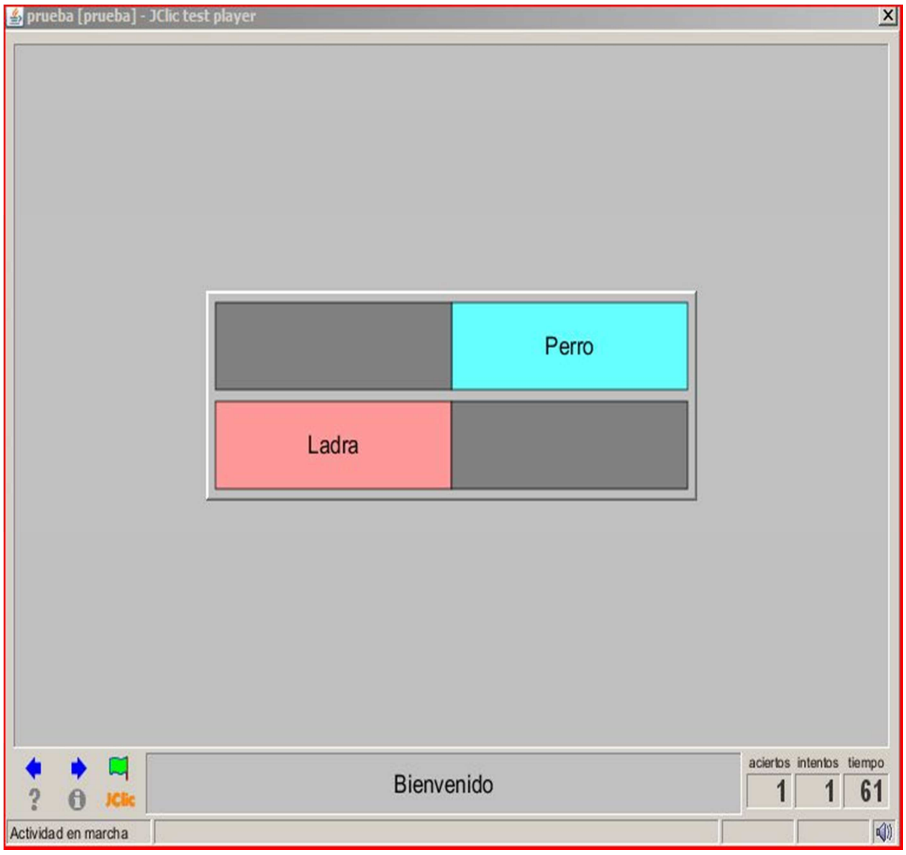

Figura 30. Pantalla de juego 2

Unity

Opiniones de la plataforma:

- Puntos fuertes:
  - Muy profesional, aporta una gran cantidad de opciones y permite realizar proyectos muy complejos.
  - Orientado a programadores.
  - Sistema de plugins basado en:
    - Escribir el plugin deseado en un lenguaje basado en C y compilarlo.
    - Crear un script C# que llame a la función del plugin para realizar la funcionalidad deseada.

  Ejemplos de extensiones que ofrece:

  - Terrain Toolkit: Integra un conjunto de herramientas para el editor Unity, las cuales facilitan la creación de terrenos realistas para los juegos.
  - Explosion Framework: Aporta efectos de explosión, pudiendo configurar su color, tamaño, duración, y la aparición de partículas como humo, ondas, etc.
  - Locomotion System: Hace que los personajes de los juegos caminen y corran por cualquier tipo de terreno irregular, de forma que parezca natural y correcto.
- Puntos débiles:
  - Es una herramienta que no es sencilla de utilizar.
  - No está orientado al e-learning.

Capturas de pantalla:

- Menú principal:

En este menú (Figura 31) disponemos de las principales funciones de la herramienta Unity. Como se puede observar en la siguiente figura el menú está dividido en diferentes paneles.

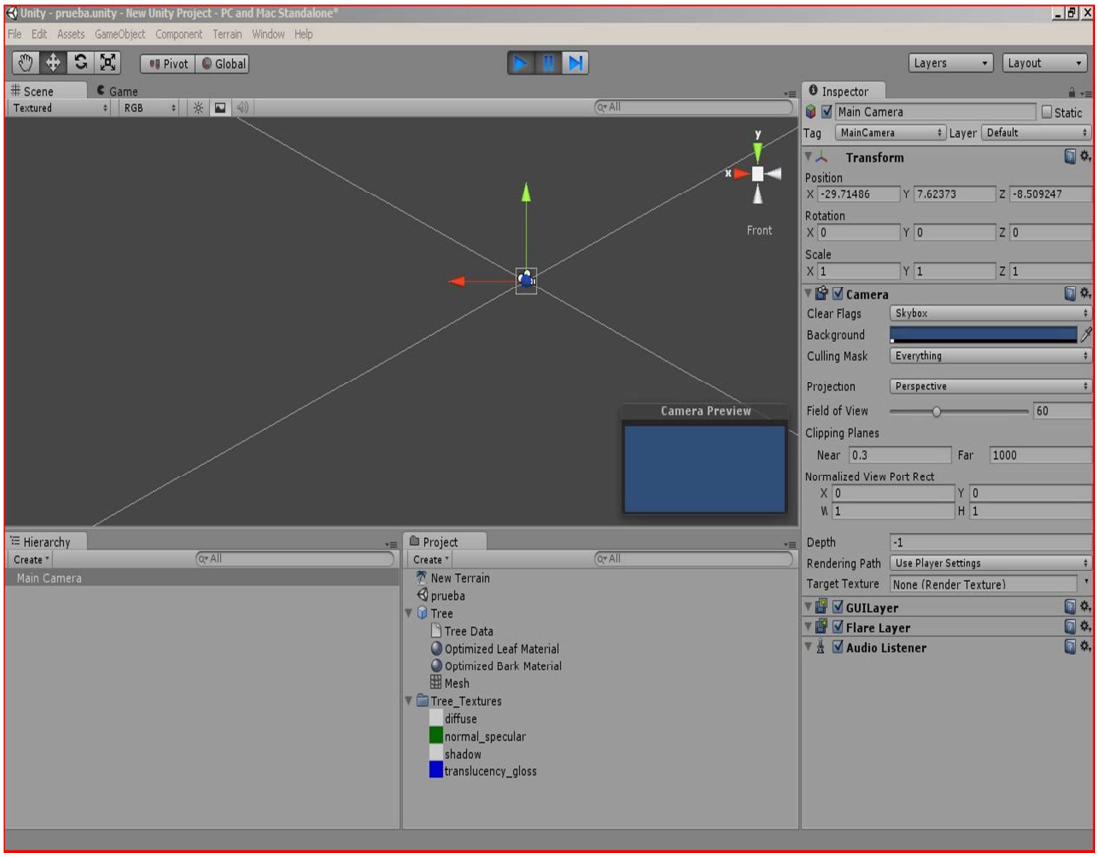

Figura 31. Menú principal Unity

- Panel de escena:

Contiene una vista gráfica de todos los objetos que forman la escena (Figura 32).

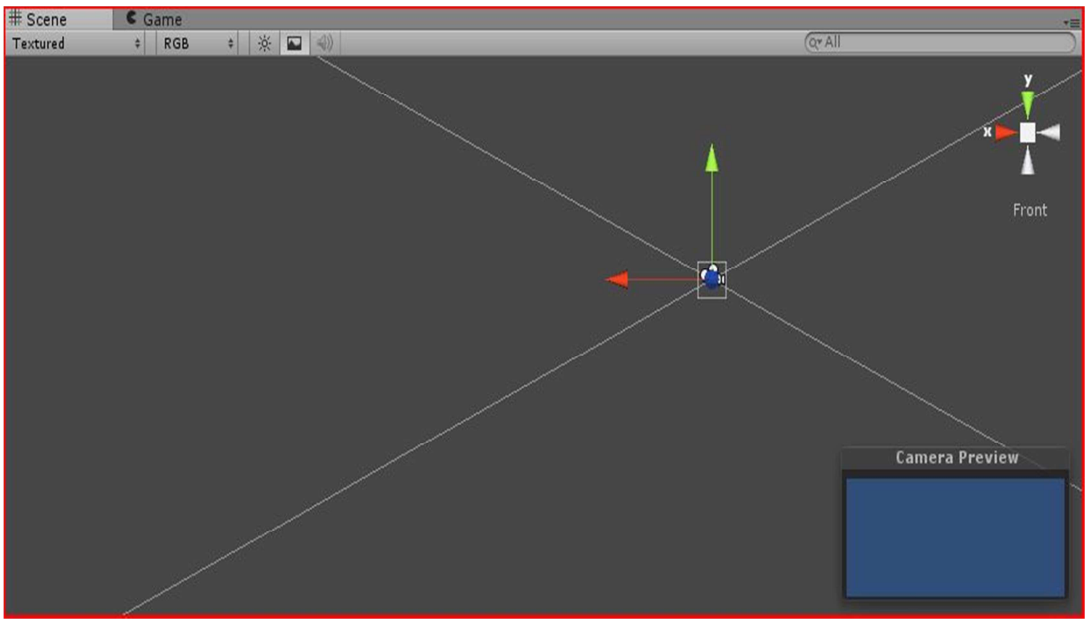

Figura 32. Panel de escena

- Panel del juego:

Se trata de una vista de previsualización del juego que estamos creando dentro del propio editor (Figura 33). Esta vista es la que ofrece la cámara principal. Este panel se puede arrastrar fuera del editor en sí en una nueva pantalla.

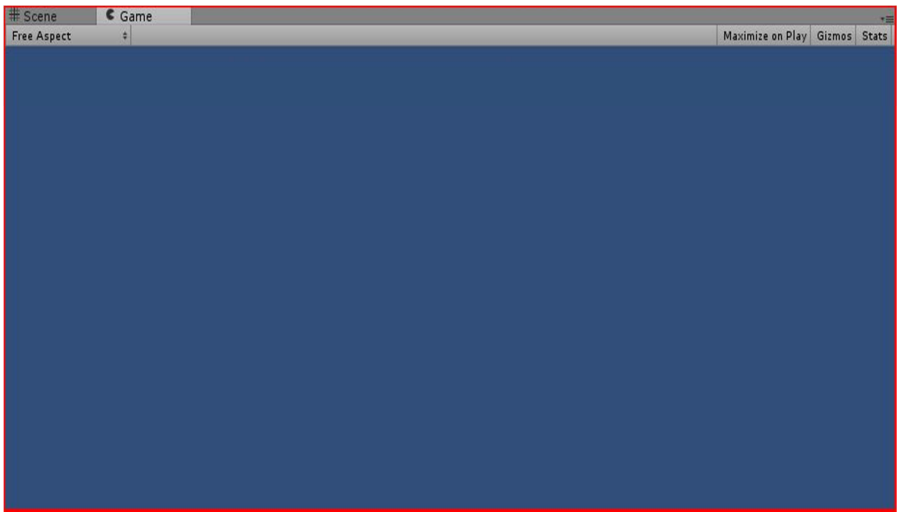

Figura 33. Panel de juego Unity

- Panel de jerarquía:

Muestra todos los objetos que componen la escena pero en forma de lista en vez de gráfica (Figura 34).

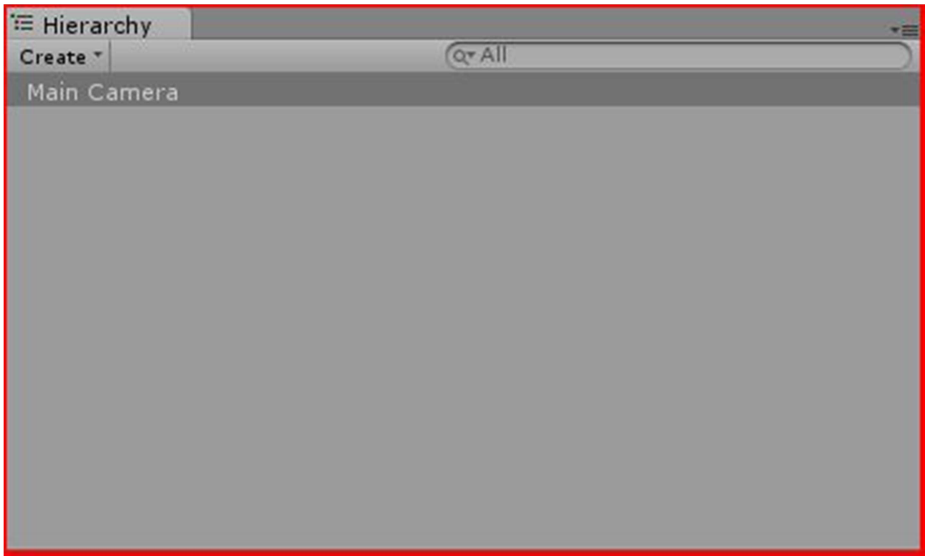

Figura 34. Panel de jerarquía de Unity

- Panel del proyecto:

Muestra todos los archivos que están implicados en este proyecto (Figura 35).

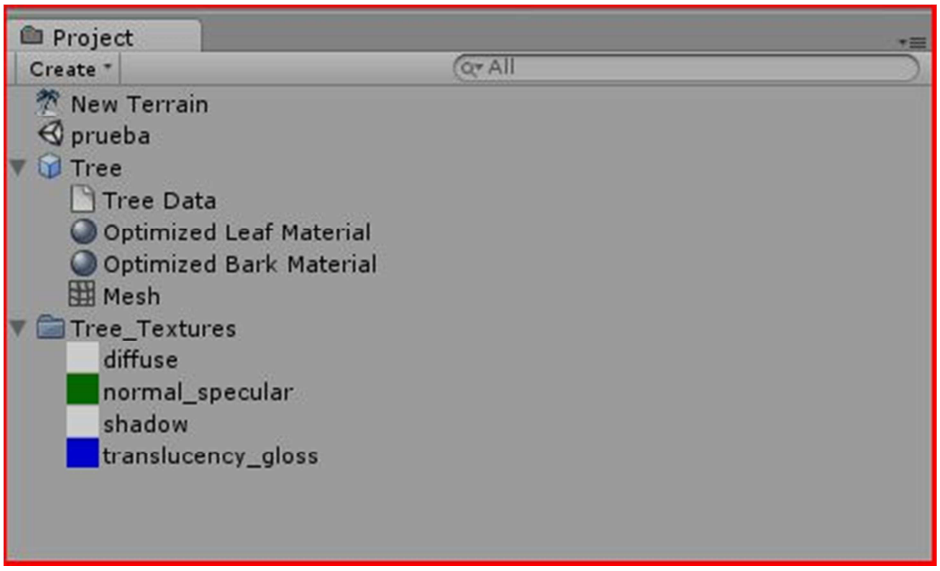

Figura 35. Panel del proyecto

- Panel "inspector"

Contiene una vista detallada del elemento seleccionado. En este caso es la cámara principal del juego (Figura 36).

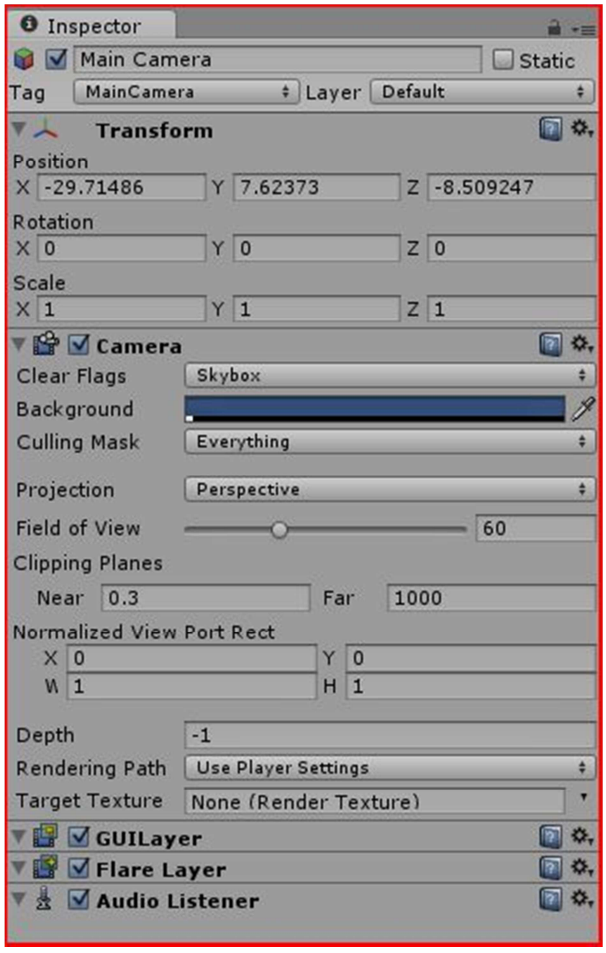

Figura 36. Panel inspector Unity

Alice

Opiniones de la plataforma:

- Puntos fuertes:
  - Tutorial nada más iniciarse el programa por primera vez.
  - Interfaz muy sencilla de usar.
  - Todo se realiza gráficamente, no hay que introducir código.
  - Plugin de NetBeans para Alice:

    Hay que bajarse los plugins de netbeans de JOGL(opengl) y el de alice. En netbeans se añaden todos estos plugins y se instalan. Tras esto, se puede importar cualquier proyecto de Alice y se generan los archivos *.java*, con lo que podemos modificar el proyecto de una manera más profesional.

- Puntos débiles:
  - Sólo creación de proyectos en 3d.

Capturas de pantalla:

- Tutorial:

Se trata de un tutorial que se ejecuta al entrar por primera vez en la aplicación (Figura 37).

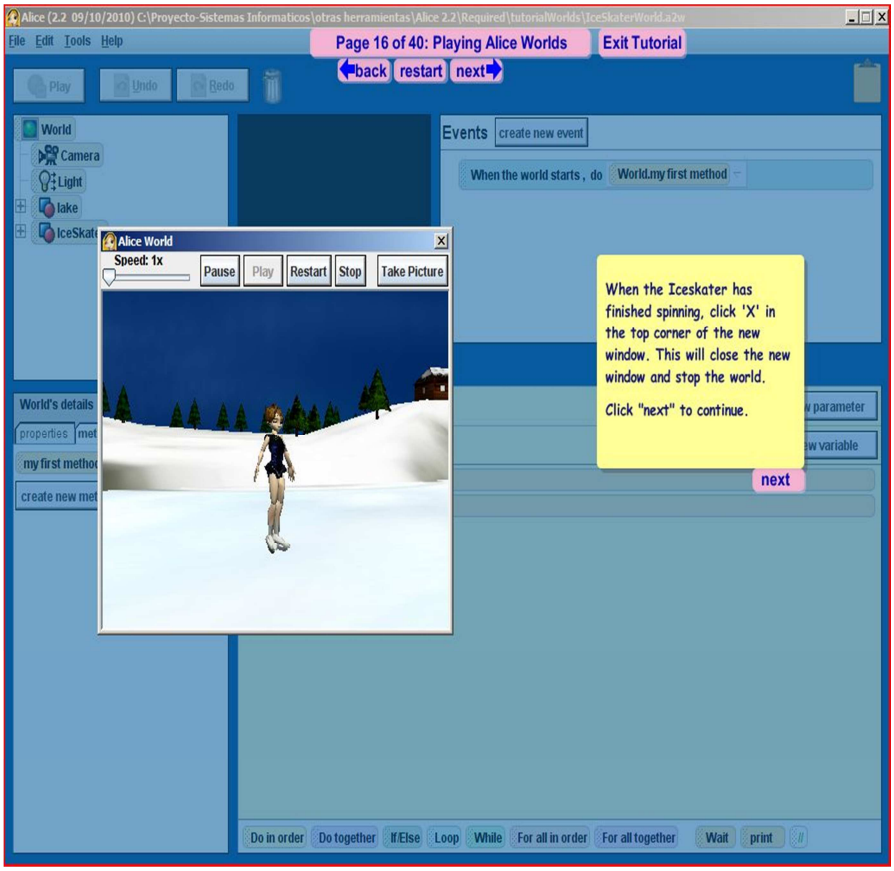

Figura 37. Imagen del tutorial de Alice

- Menú principal:

En la siguiente figura (Figura 38) podemos observar el menú principal de la aplicación. En esta se pueden realizar la gran mayoría de las acciones que realizaremos en la creación de las animaciones o juegos de Alice.

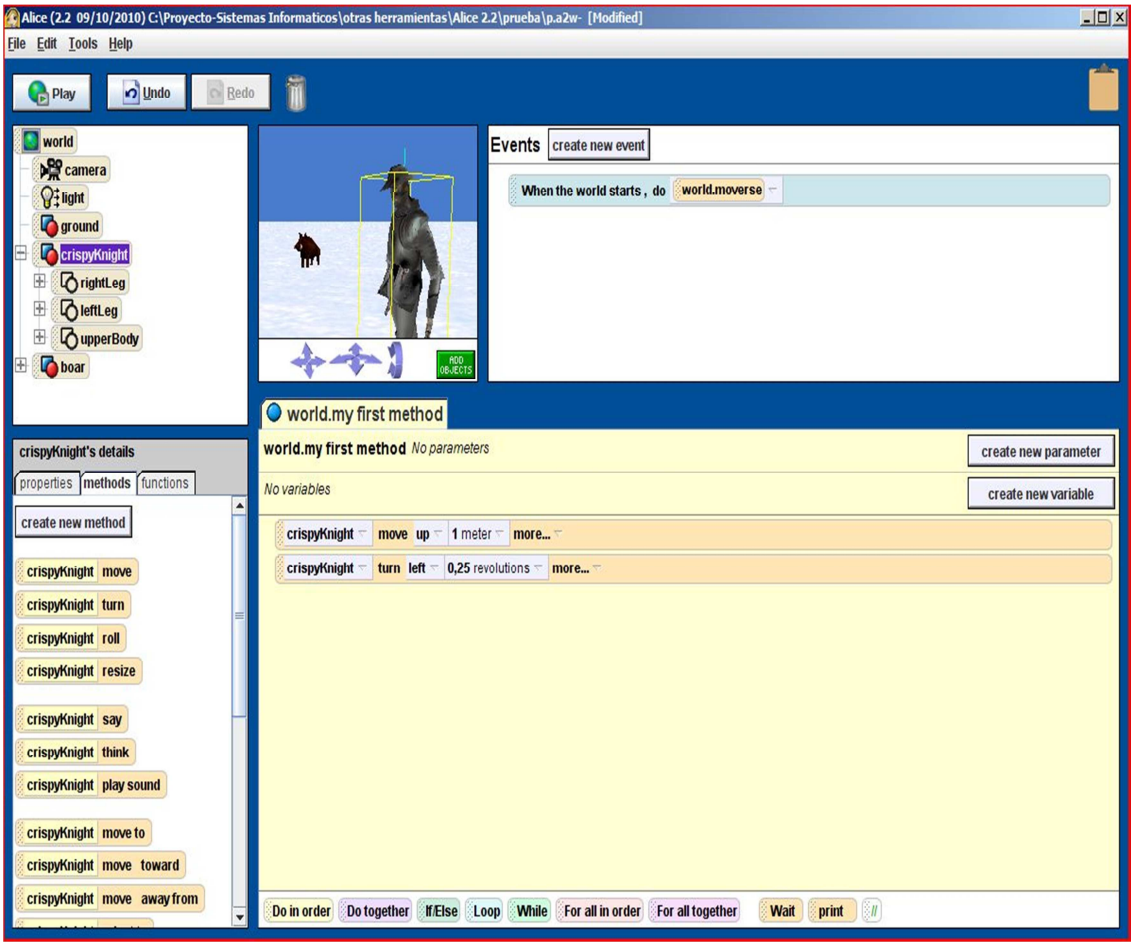

Figura 38. Menú principal Alice

Se pueden observar ciertos paneles secundarios dentro de este panel principal:

- Árbol de elementos

Aquí disponemos de todos los objetos de la animación, esto es, los objetos que forman parte de la animación en sí, como el fondo y la cámara (Figura 39).

Figura 39. Árbol de elementos

- Propiedades del elemento seleccionado

En este panel (Figura 40) podemos comprobar las distintas funciones, propiedades y métodos del elemento seleccionado. Para hacer uso de ellos tan sólo tenemos que arrastrar el deseado al panel de ejecución.

Figura 40. Propiedades de un elemento

- Previsualización de la animación creada (Figura 41)

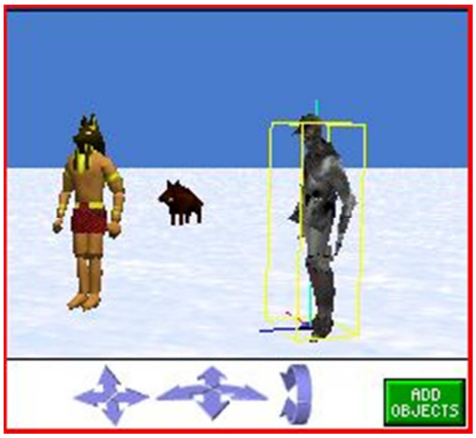

Figura 41. Previsualización

- Panel de ejecución

Aquí es donde se determina el orden y lo que se ejecuta en el proyecto creado. Podemos comprobar que en la parte inferior disponemos de ciertas estructuras genéricas como son "while", "loop" o "if" (Figura 42).

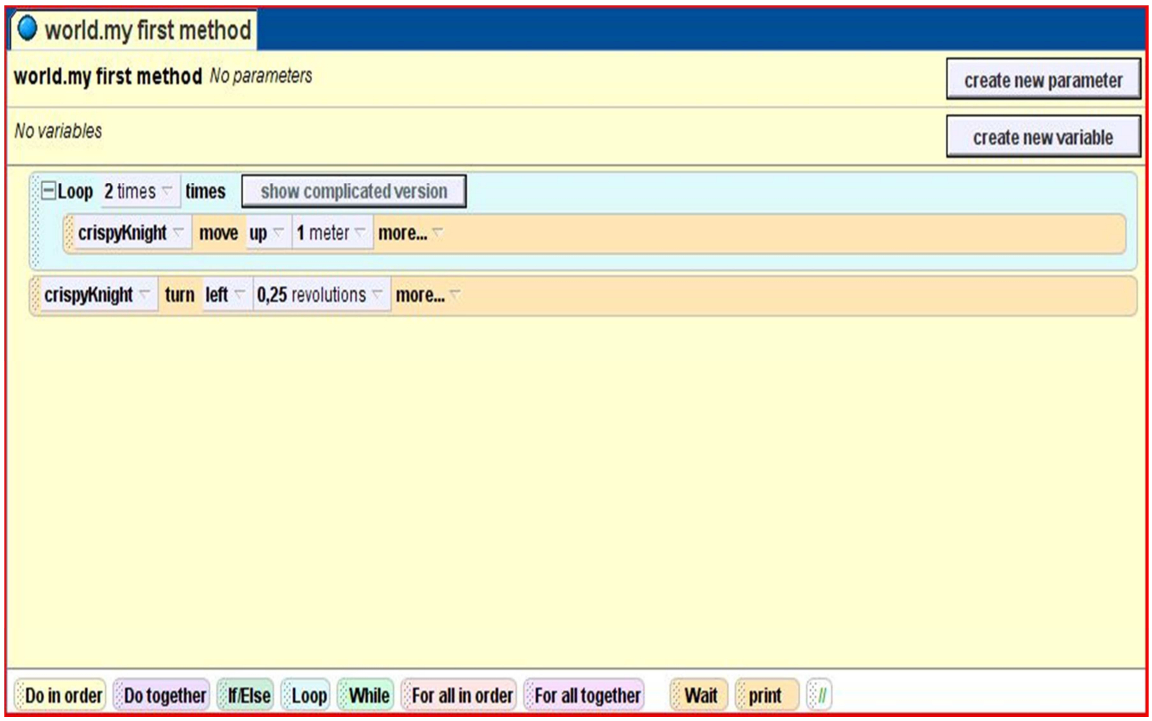

Figura 42. Panel de ejecución

- Añadir objetos

Para añadir un nuevo objeto al proyecto, tan sólo debemos pulsar el botón correspondiente y elegir entre las galerías por defecto o cargar los objetos propios que hayamos creado (Figura 43).

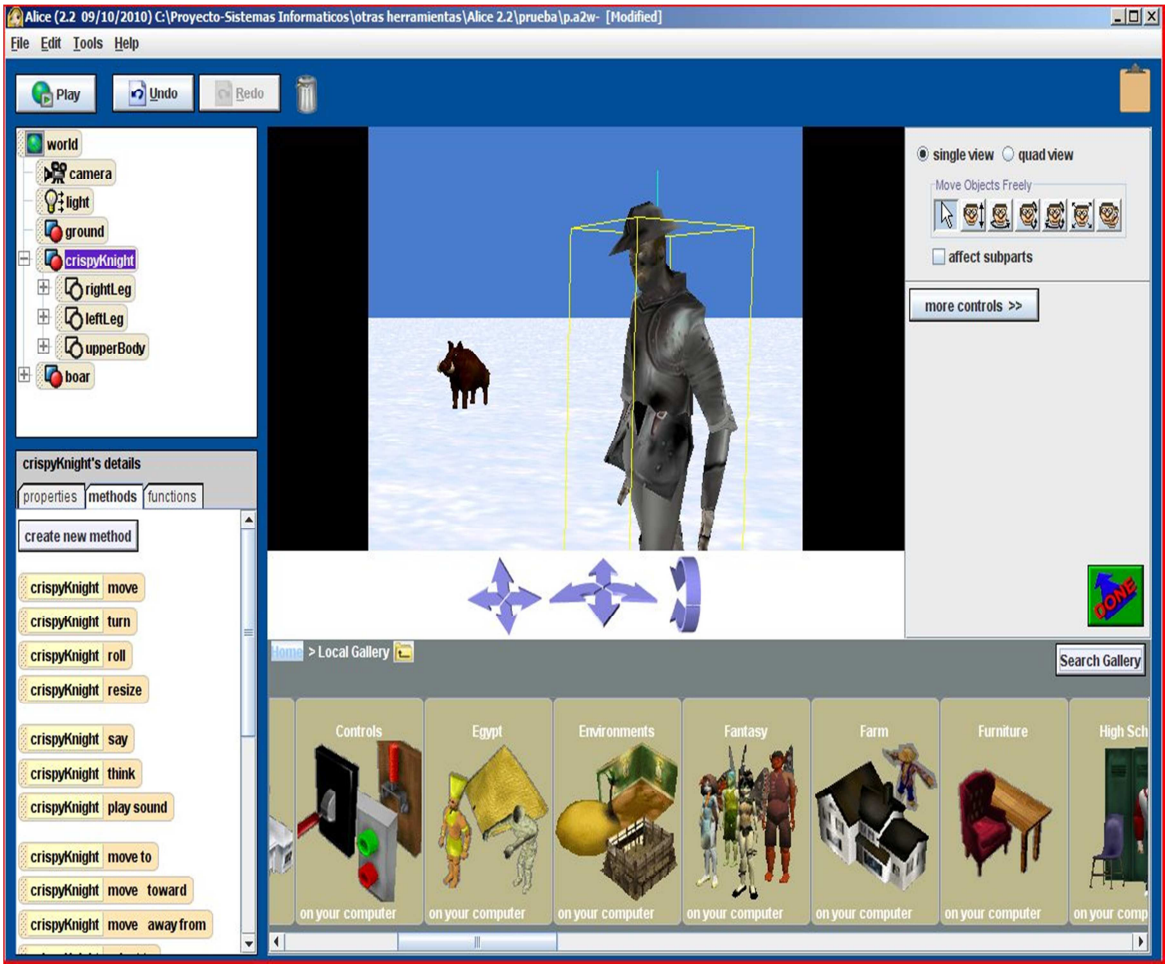

Figura 43. Ejemplo de selección de un objeto

Raptivity

Opiniones de la herramienta:

- Puntos fuertes:
  - Sencillo de usar.
  - Visualmente atractivo.
  - Múltiples tipos de proyectos a realizar. Los ofrecen como base para que puedan ser modificados con lo que interese hacer.
  - Tiene un acabado muy profesional.
- Puntos débiles:
  - No es posible realizar proyectos no basados en los tipos que ofrecen.
  - Herramienta muy pesada.
  - No parece ser una herramienta pensada para realizar grandes proyectos sino pequeñas pruebas.

Capturas de pantalla:

- Menú principal:

En este menú (Figura 44) disponemos de las principales funciones de la herramienta Raptivity. Como se puede observar en la figura, el menú está dividido en diferentes paneles en los que se puede elegir entre crear una nueva interacción (actividad), abrir una creada anteriormente, buscar en la ayuda de la herramienta, etc.

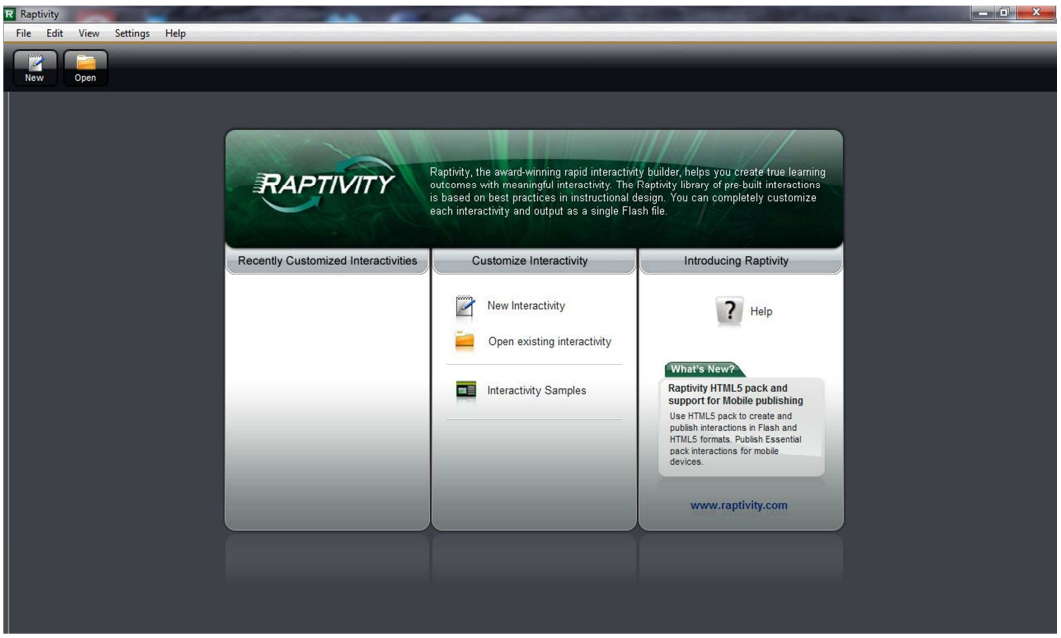

Figura 44. Menú principal

- Tipos de interacciones:

En esta imagen (Figura 45) podemos observar el tipo de actividades de que dispone Raptivity para el pack "esencial". Está dividido en diferentes temas desde encuestas a presentaciones, diagramas interactivos, etc. Otros packs de la herramienta ofrecen diferentes tipos de actividades.

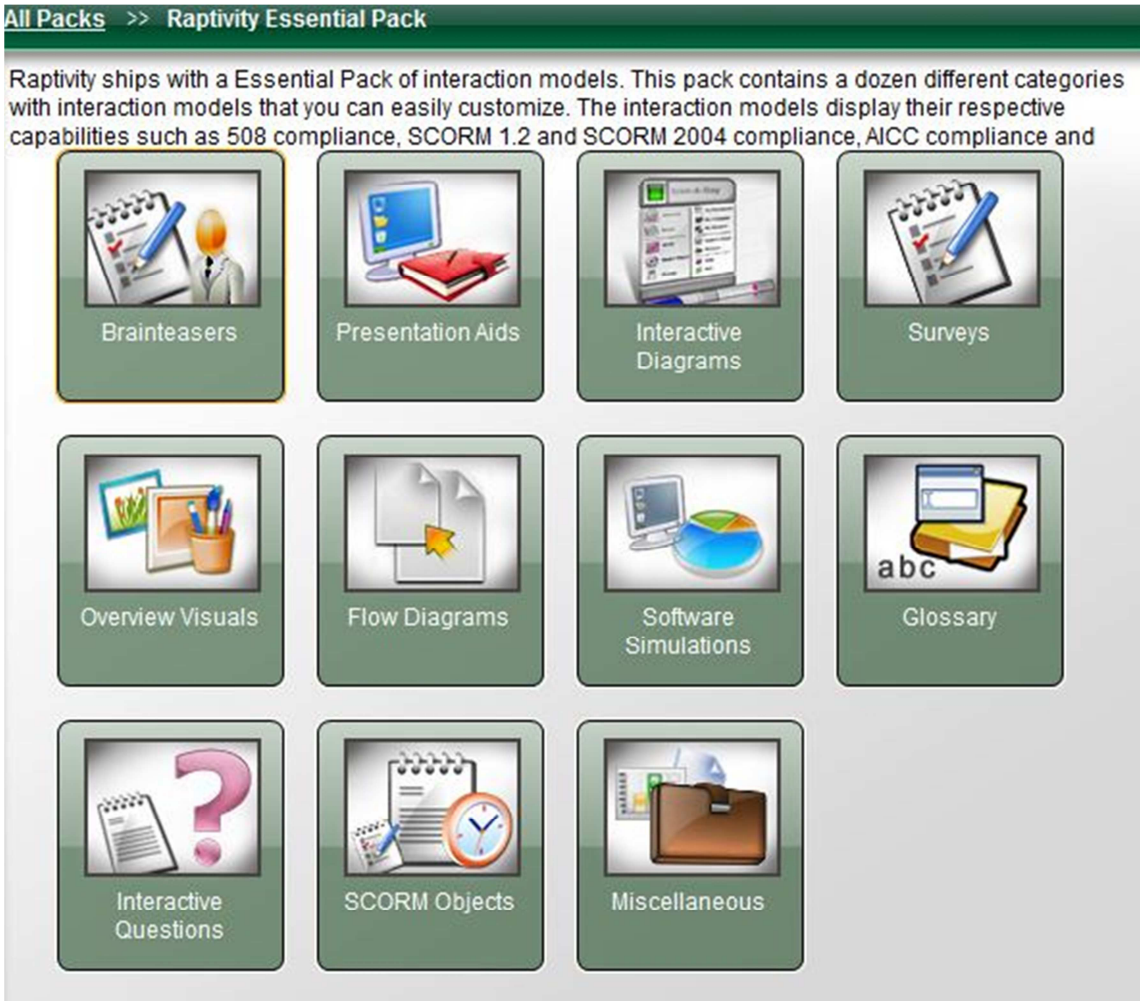

Figura 45. Tipos de actividades (interacciones) del pack "esencial"

- Actividades modelo:

Raptivity incluye una serie ejemplos de actividades ya realizadas y que pueden manipularse fácilmente, aplicando los cambios que consideremos necesarios para adaptarlo al propio interés.

Como ejemplo, podemos observar (Figura 46 y Figura 47) una presentación interactiva de importantes astrónomos. En ella podemos pinchar sobre cada una de las cartas. Esto provoca que la carta gire y nos muestre, en el reverso, la contribución del astrónomo en cuestión.

Figura 46. Presentación interactiva con cartas

Figura 47. Presentación interactiva con cartas 2

- Actividades propias:

Al crear una actividad propia, disponemos de un tutorial inicial bastante completo que muestra las diferentes opciones que se pueden realizar. Como ejemplo de actividad hemos elegido crear una encuesta.

El tutorial para esta “interacción” se puede contemplar en la figura (Figura 48).

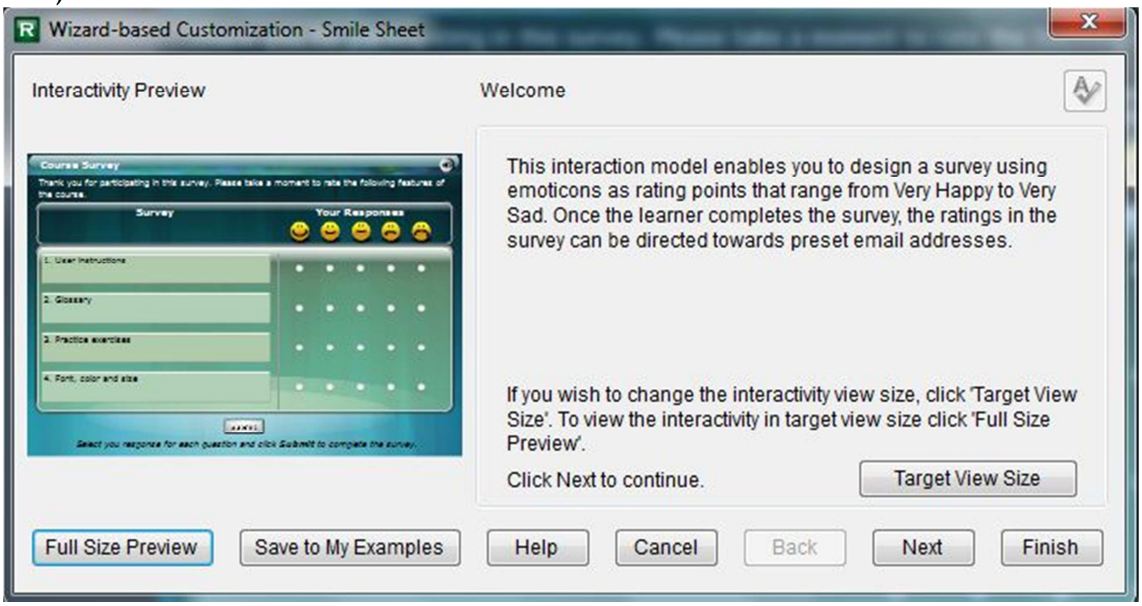

Figura 48. Tutorial de encuesta

Tras el tutorial, se pueden personalizar cada uno de los campos de la actividad, desde las instrucciones de la encuesta, al tamaño, etc (Figura 49).

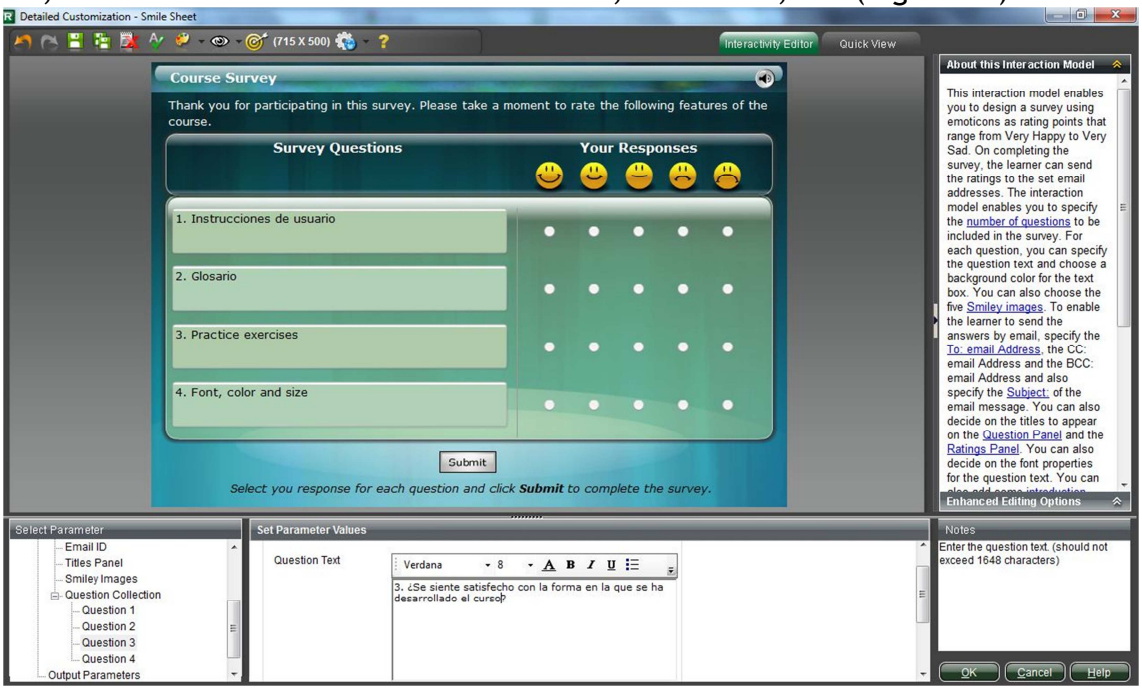

Figura 49. Personalización de la actividad

Scratch

Opiniones de la herramienta:

- Puntos fuertes:
  - Interfaz gráfica muy visual y sencilla.

- No hace falta programar sino que las distintas opciones están formadas a modo de “piezas” que pueden encajarse fácilmente entre ellas.
- Dispone de muchos recursos para utilizar: animaciones, imágenes, sonidos, fondos, etc.
- Contiene muchas funciones posibles incluyendo el uso de variables, sensores, acciones condicionadas, etc.
- Ofrece a parte de poder importar nuestros propios recursos de una herramienta tipo Paint para dibujarlo directamente sobre Scratch.

- Puntos débiles:
  - Tal vez demasiado orientado a niños lo que no permitiría proyectos muy complejos o se harían difíciles de mantener.

Capturas de pantalla:

- Menú principal:

A continuación podemos observar el menú principal de la aplicación (Figura 50). En esta pantalla se pueden realizar todas las acciones para crear un nuevo juego o animación.

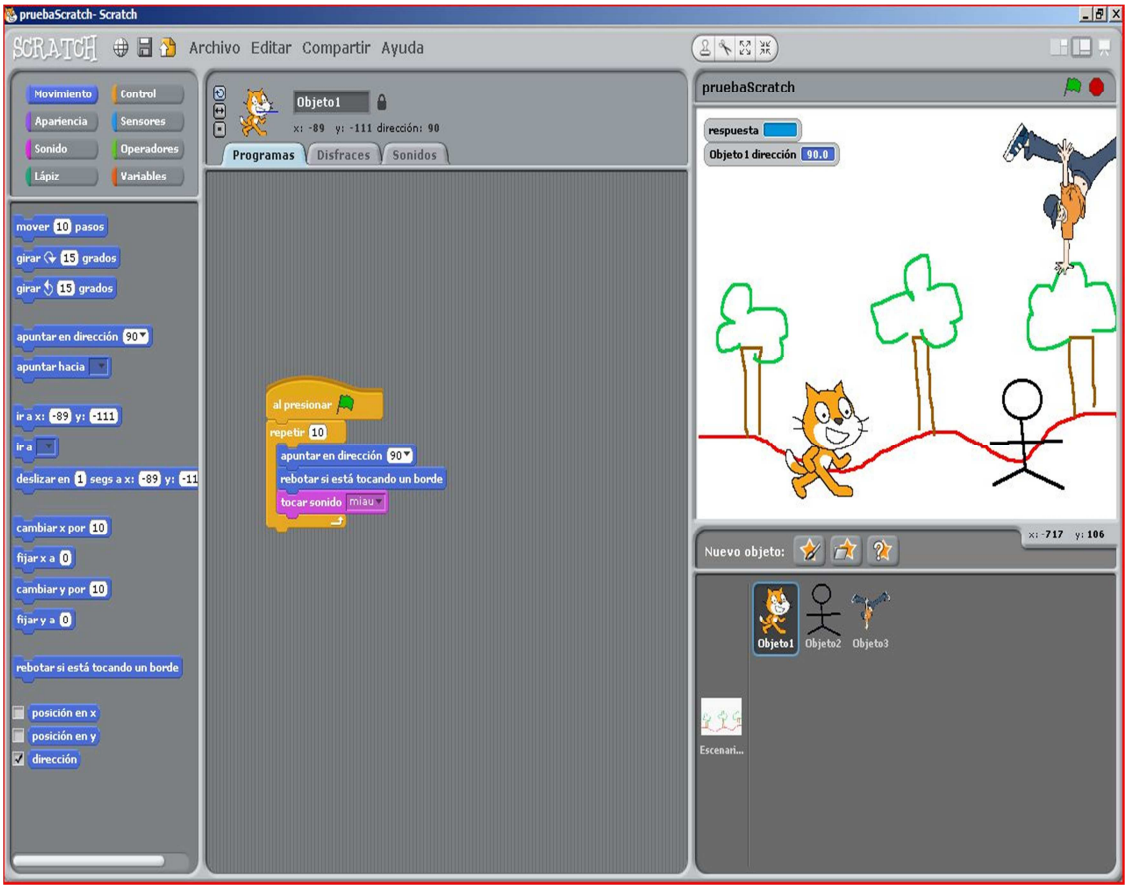

Figura 50. Menú principal de Scratch

Como podemos comprobar, el menú principal está dividido en varios paneles donde realizaremos diferentes operaciones:

- Añadir escenario:

Se trata del típico sistema de archivos donde podemos elegir un escenario ya incluido en la herramienta, uno propio o pintarlo directamente (Figura 51).

Figura 51. Selección de escenario

- Añadir objeto:

Análogo a la inserción de un escenario (Figura 52).

Figura 52. Selección de objeto

- Funciones del elemento

En este panel (Figura 53) se eligen las operaciones que queremos que realize nuestro objeto seleccionado. Existen varias categorías que incluyen diferentes funciones según la misma.

Figura 53. Panel de funciones de un objeto

- Tareas de ejecución

Aquí es donde se definen las acciones que va a realizar cada uno de nuestros objetos. Tan sólo tenemos que arrastrar al panel las funciones que deseemos desde el panel de funciones del objeto seleccionado.

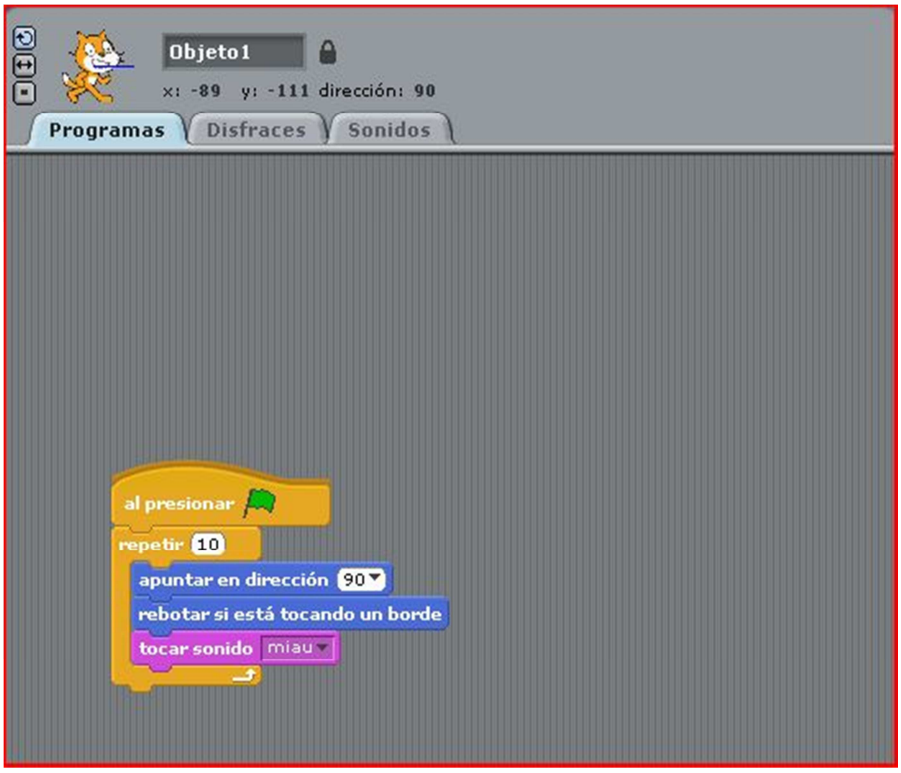

Figura 54. Imagen de las tareas de ejecución

- Visualización:

En este apartado podemos comprobar el resultado final. Se puede elegir el tamaño de la pantalla a la hora de ejecutarlo.

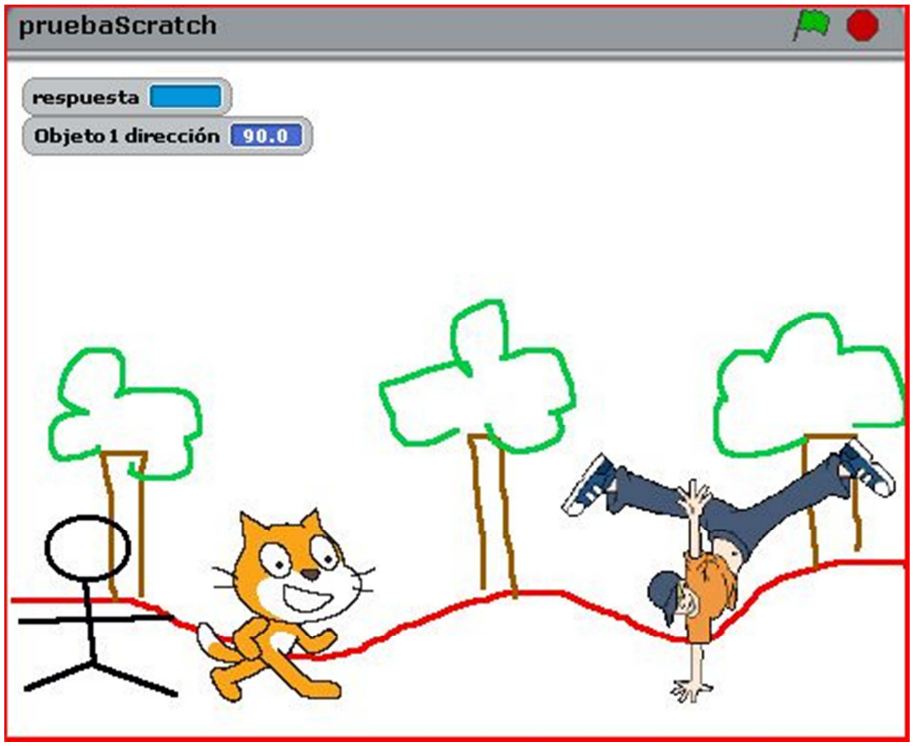

Figura 55. Ejemplo de visualización

3.2 Plataformas para la gestión de plugins

A continuación vamos a analizar las diferentes plataformas que nos permiten integrar un sistema de plugins en nuestros desarrollos centrándonos en las que están orientadas al lenguaje Java, ya que es el lenguaje utilizado en la plataforma e-Adventure.

En algunas de estas plataformas nos hemos ayudado de un pequeño ejemplo para poder evaluar de primera mano algunos aspectos como la curva de aprendizaje en la plataforma en cuestión, la facilidad de adaptar una aplicación ya existente, la facilidad de crear un plugin, etc.

El ejemplo en cuestión es un editor de texto sencillo, que únicamente constará de una caja de texto y dos botones, uno con funcionalidad de edición (buscar en el texto, cortar o copiar el texto seleccionado y pegar el contenido del portapapeles en el editor) y otro con funcionalidad asociada a ficheros (abrir el contenido de un fichero de texto en el editor, guardar el contenido del editor en un fichero de texto y salir del editor).

El objetivo es conseguir adaptar el ejemplo como plataforma de plugins y separar la funcionalidad del botón de edición y utilizarlo como plugin que instalemos en nuestro editor.

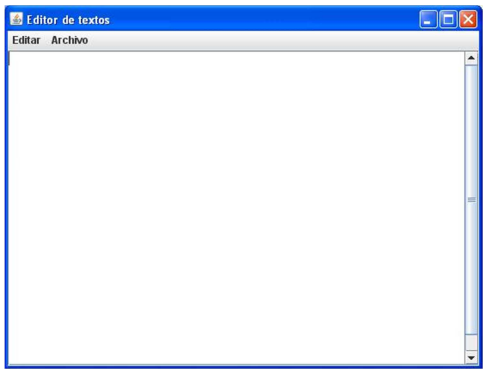

Figura 56. Imagen genérica de editor de texto

a) OSGI: Open Services Gateway Initiative[29]

OSGI proporciona un marco de trabajo para aplicaciones java de uso general, seguro y administrado que soporta el despliegue dinámico de aplicaciones conocidas como Bundles o módulos. Fue creado en 1999 por OSGI Alliance, un consorcio de empresas tecnológicas a nivel mundial (entre ellas Motorola, Nokia, LG Electronics o Vodafone) y fue pensado principalmente para su aplicación en redes domésticas.

29 OSGI: http://www.osgi.org/Main/HomePage

Algunas de las plataformas certificadas en OSGI son:

- Apache Felix[30]

  Es una implementación de OSGI R4 que inicialmente partió del código fuente donado por el proyecto OSCAR de ObjectWeb.

- Eclipse Equinox[31]

  Plataforma que implementa la R4 de OSGI. Es núcleo sobre el que se basa el IDE de desarrollo Eclipse.

- Knopflerfish[32]

  Plataforma de servicios OSGI creado y mantenido por Makewave.

- FUSE ESB 4[33]

  Proyecto OpenSource basado en Apache ServiceMix 4 y que tiene una implementación distribuida de OSGI R4.

  Algunos de los beneficios que proporciona esta tecnología son:

  - Reutilización del código "out of the box".
  - Simplifica los proyectos en los que participan muchos desarrolladores en diferentes equipos.
  - Se puede utilizar en sistemas más pequeños.
  - Gestiona los despliegues locales o remotos.
  - Se trata de una herramienta fácilmente ampliable.
  - No es una tecnología cerrada.
  - Gran aceptación. Es una tecnología usada por muchas empresas fabricantes.

Para entender mejor este framework, pasamos a analizar los elementos más importantes: Bundles y Arquitectura OSGI.

BUNDLES

Un bundle es una aplicación empaquetada en un fichero jar que se despliega en una plataforma OSGI. Cada uno contiene un fichero de metadatos que se llama MANIFEST.MF, que se encuentra ubicado en un directorio META-INF y que está organizado por pares clave-valor donde describen la relación del bundle con el resto del entorno, por ejemplo:

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.7.0
```

30 Apache Felix http://x.apache.org/site/index.html
31 Eclipse Equinox http://www.eclipse.org/equinox/
32 Knopflerfish http://www.knopflerfish.org/
33 FUSE ESB 4 http://fusesource.com/products/enterprise-servicemix/

```
Created-By: 10.0-b22 (Sun Microsystems Inc.)
Bundle-ManifestVersion: 2
Bundle-Name: Datastorage Plug-in (5)
Bundle-SymbolicName: com.javahispano.test
Bundle-Version: 1.0.0
Bundle-Activator: com.javahispano.test.Activator (2)
Bundle-Vendor: Javahispano
Eclipse-LazyStart: true (1)
Import-Package: org.osgi.framework;version="1.3.0"  (3)
Bundle-ClassPath: lib/milibreria.jar, lib/milibreria2.jar (4)
```

En él se definen algunas directivas que definirán el comportamiento del bundle, como por ejemplo:

(1) La política de activación del bundle (indicando si se cargará siempre o únicamente cuando es requerido por otro bundle).
(2) La clase que será invocada en el momento de activación del bundle.
(3) La categoría (nombre con el que podemos agrupar bundles).
(4) Las rutas de librerías dentro del jar.
Las características del bundle: nombre (5), descripcion, URL de documentación, dirección de contacto, etc.

El desarrollador se encargará de controlar el ciclo de vida del bundle, implementando una serie de interfaces proporcionadas por OSGI. La principal interfaz es org.osgi.framework.BundleActivator, en la que tenemos que implementar dos métodos: Start y Stop.

ARQUITECTURA

OSGI tiene un modelo en capas que se representa en la siguiente figura (Figura 57).

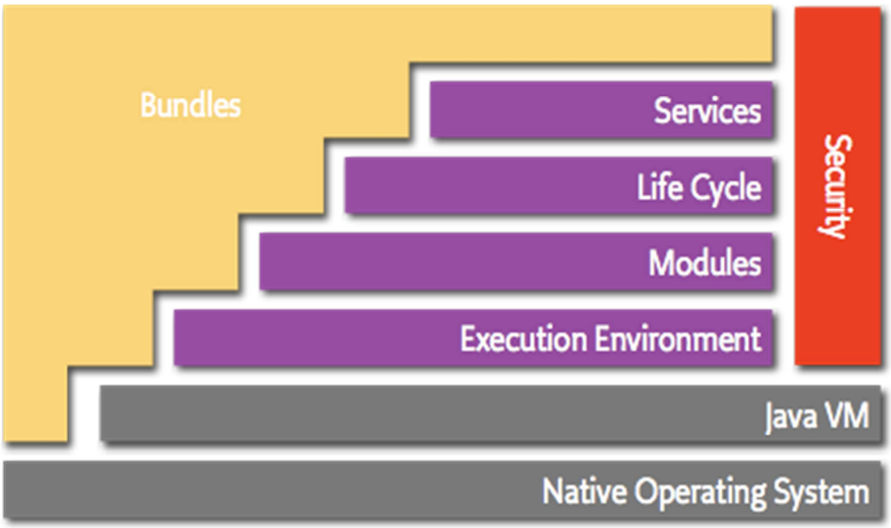

Figura 57. Imagen de las diferentes capas OSGI[34]

34 http://www.osgi.org/Main/HomePage

A continuación describiremos brevemente cada capa del modelo OSGI:

SECURITY LAYER

Es una capa opcional basada en Java 2 Security que provee una infraestructura para desplegar y manejar aplicaciones que tienen que ejecutarse en un entorno seguro y controlado. OSGI proporciona un modelo de seguridad capaz de autenticar el código a través de la ubicación y firma del bundle y sus paquetes. Así, la finalidad de esta capa la podríamos resumir en gestionar la integridad y la autenticación tanto del bundle como de sus paquetes.

MODULE LAYER

OSGI proporciona una solución genérica y estandarizada para la modularización de aplicaciones java, y esta capa define la anatomía de los bundles, las reglas de interoperación entre ellos y la arquitectura de carga de clases.

LIFE CYCLE LAYER

Esta capa nos permite gestionar el ciclo de vida de un bundle, controlando los distintos estados por los que éste puede pasar, que son los siguientes:

- Installed: El bundle ha sido instalado correctamente.
- Resolved: Cuando todas las clases java que el bundle necesita están disponibles. En este estado el bundle está listo para ser arrancado o ha sido detenido.

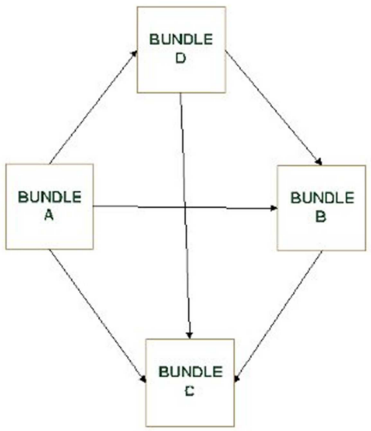

Figura 58. Relación de los bundles en la plataforma[35]

35 Pagina tutorial de Roberto Montero:
http://www.javahispano.org/contenidos/archivo/14922/OSGI_Roberto_Montero.pdf

- Starting: El bundle está arrancando. En caso de que hayamos establecido la propiedad una política de activación perezosa, se quedará en este estado hasta que sea necesario su arranque.
- Active: El bundle ha sido correctamente activado y está en funcionamiento.
- Stopping: El bundle ha sido detenido.
- Uninstalled: El bundle ha sido desinstalado.

En la siguiente figura (Figura 59) se muestra el ciclo de vida de un bundle en OSGI.

SERVICE REGISTRY LAYER

Un servicio en OSGI es un objeto java que es registrado bajo una o varias interfaces. Los bundles pueden registrar servicios, buscarlos o recibir notificaciones cuando el estado de un servicio sufre alguna variación.

A través del contexto del bundle podemos podremos registrar una clase java como un servicio, buscar servicios registrados en la plataforma y escuchar y filtrar los eventos producidos en los cambios de estado de un servicio.

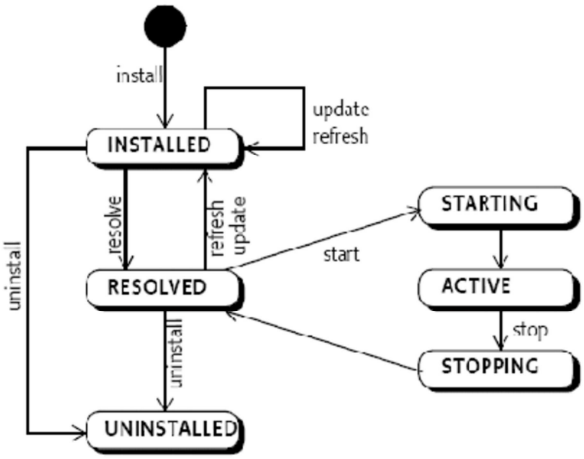

Figura 59. Ciclo de vida de un bundle

Además de los servicios registrados por el desarrollador, la plataforma proporciona una serie de servicios, como los de logging, servidor HTTP y el Device Access Specification o DAS, que permite el descubrir los dispositivos y servicios ofrecidos por éstos su implementación.

Observaciones

OSGI fue la primera plataforma que descartamos como solución al proyecto que nos ocupa, ya que consideramos que era demasiado compleja para nuestras necesidades y además, ofrecía mucha funcionalidad que no íbamos a aprovechar.

Una de las características que más influyó en esta decisión fue que debíamos reestructurar completamente el código de e-Adventure en módulos

OSGI para luego hacerlo funcionar sobre alguna de las plataformas certificadas como Eclipse Equinox, Apache Félix etc. y esto no compensaba respecto al provecho que íbamos a sacar de ella.

b) JPF: Java Plugin Framework[36]

JPF proporciona un motor que gestiona de forma dinámica la carga y ejecución de plugins. Mantiene un registro de plugins disponibles y de las funciones que ofrecen a través de puntos de extensión y extensiones.

Las principales ventajas de esta arquitectura son:

- Arquitectura abierta: su API está diseñada como un conjunto de clases abstractas e interfaces Java, por lo que cada diseñador puede tomar sus propias decisiones acerca del desarrollo específico para su aplicación. También provee una implementación "por defecto".
- Diseño del API consistente y limpio: han tenido especial cuidado en reducir el tiempo que necesite un desarrollador para familiarizarse con el framework.
- Chequeo de integridad de plugins al arranque del motor.
- Los plugins pueden incluir documentación propia, tanto dentro del propio fichero "manifest" como referencias a documentos incluidos dentro del plugin.
- Chequeo de dependencias entre plugins.
- Activación perezosa de plugins: éstos son cargados en memoria únicamente si son necesarios.
- Hay 2 tipos de funcionalidad en los plugins Java Plugin Framework:
    - Modularidad: Dividir la aplicación en partes más pequeñas. Por ejemplo: Suponemos que un plugin cuenta con una clase A (esta clase es incluida en un directorio jerárquico incluido en el manifest del plugin). Si queremos ahora incluir otro plugin con una clase B que tenga dependencia con respecto a A, hay que incluir en el manifest de B una entrada que ponga que el plugin B depende de A. Simplemente introduciendo esta entrada en el manifest la clase A será visible para el código de la clase B.
    - Extensibilidad: La extensión de la aplicación se añade a una funcionalidad ya existente. Esto se desarrolla con declaraciones especiales dentro del archivo manifest. Por ejemplo: En JPF extensibilidad se basa en el concepto de puntos de extensión y las extensiones. Una extensión es código que se añade a un punto de extensión. Normalmente los puntos de extensión son declarados en el manifest del plugin e implementados con código Java.

Observaciones

- La documentación es bastante escasa. Disponen de un sólo plugin de ejemplo.

36 JPF http://jpf.sourceforge.net/index.html

- Los plugins JPF presentan un pequeño problema de compatibilidad con eclipse y es que el manifest de los plugins JPF es diferente al usado por el IDE, por lo que hay que adecuarlo.
- Eclipse está orientándose a un modelo de plugins OSGI y está haciendo obsoleto los plugins basados en XML.
- Estructura de un plugin JPF (siguiendo el ejemplo de la página oficial):

El núcleo de los plugins está formado por dos partes:

1. El "manifest"

El manifest del plugin es un archivo con sintaxis XML que ha de crearse siguiendo una estructura claramente definida.

La primera parte del manifest es el tag raíz del XML:

```
<plugin id="org.jpf.demo.toolbox.core" version="0.0.4"
        class="org.jpf.demo.toolbox.core.CorePlugin">
```

Aquí podemos comprobar que el ID del plugin es "org.jpf.demo.toolbox.core" y el identificador de versión "0.0.4". También se declara que el plugin tiene una clase plugin: "org.jpf.demo.toolbox.core". Esta clase es un elemento opcional, no obstante.

A continuación, se declaran las librerías que van a ser necesarias para la ejecución del plugin:

```
<runtime>
   <library id="core" path="classes/" type="code">
      <export prefix="*"/>
   </library>
   <library type="resources" path="icons/" id="icons">
      <exportprefix="*"/>
   </library>
</runtime>
```

Aquí definimos que todo el código Java del plugin se emplazará en la carpeta "classes/" dentro del contexto de la carpeta "home".

También declaramos que todas las clases y paquetes de este plugin son visibles a otros plugins de tal manera que el código pueda ser reutilizado libremente. Además, se define una carpeta de recursos "icons\" y lo hace visible a otros plugins.

La última parte del "manifest" es la más importante, porque es la que hace que el pluginsea lo más extensible posible. Esta es la declaración de extensiones:

```
<extension-point id="Tool">
   <parameter-def id="class"/>
   <parameter-def id="name"/>
   <parameter-def id="description" multiplicity="none-or-one"/>
   <parameter-def id="icon" multiplicity="none-or-one"/>
</extension-point>
```

Con este código se está declarando que nuestro plugin "expone" un punto donde puede ser extendido por cualquier otro plugin. Este punto se denomina "Tool" y explica que la extensión de este punto se representará como una pestaña en la GUI de la aplicación.

Cualquier plugin que contribuya a este punto de extensión debe proveer diferentes parámetros que serán usados para aplicar el plugin a la aplicación y comunicarse con él. Se definen cuatro parámetros para este punto de extensión:

- class: Este es un parámetro obligatorio de tipo "String" que debe contener el nombre completo de la clase Java.
- name:El nombre de la herramienta ("tool") que será mostrado como nombre de la pestaña en el GUI.
- description:La descripción de la herramienta que será mostrada como ayuda en el GUI. Se trata de un parámetro opcional.
- Icon: Otro parámetro opcional. Se trata de la ruta al recurso con el icono de la herramienta.

2. El código de la aplicación

Hemos declarado en el "manifest" que proveeríamos una clase "org.jpf.demo.toolbox.core.CorePlugin". Normalmente, se tiene que extender la clase abstracta de JPF "org.java.plugin.Plugin" e implementar dos métodos:

"protected void doStart() throws Exception"
"protected void doStop() throws Exception"

La principal tarea de este plugin es crear y mostrar una aplicación GUI. También se quiere implementar el soporte de la lógica para los puntos de extensión definidos en el "manifest".

El principal truco está organizar eficientemente la lógica de la GUI. El principio fundamental es activar los otros plugins lo más tarde posible y tomar la máxima información posible de los puntos de extensión del archivo "manifest".

La parte más interesante del código es la comunicación con el "framework" del plugin para obtener todas las extensiones que están conectadas a nuestro punto de extensión:

```
ExtensionPointtoolExtPoint =
   getManager().getRegistry().getExtensionPoint(
      getDescriptor().getId(), "Tool");
for (Iterator it = toolExtPoint.getConnectedExtensions()
      .iterator(); it.hasNext();) {
   Extension ext = (Extension) it.next();
   JPanel panel = new JPanel();
   panel.putClientProperty("extension", ext);
   Parameter descrParam = ext.getParameter("description");
   Parameter iconParam = ext.getParameter("icon");
   URL iconUrl = null;
   if (iconParam != null) {
      iconUrl = getManager().getPluginClassLoader(
         ext.getDeclaringPluginDescriptor())
            .getResource(iconParam.valueAsString());
   }
   tabbedPane.addTab(
      ext.getParameter("name").valueAsString(),
      (iconUrl != null) ? newImageIcon(iconUrl) : null,
      panel, (descrParam != null) ?
         descrParam.valueAsString() : "");
}
```

Lo siguiente que toma lugar aquí es el contrato que se está definiendo para la clase extensión.Aquí se establece el parámetro “class”, especificado en la declaración de extensiones que debe referir a una clase Java que implementala interfaz “org.jpf.demo.toolbox.core.Tool”, definida en nuestro plugin.

También se contempla que los objetos de esta clase se instanciarán usando el constructor vacío por defecto. Se realiza, además, el compromiso de que el método “init” será llamado al menos una vez durante el ciclo de vida de la extensión. A continuación se muestra el código que implementa el concepto:

```
// Activate plug-in that declares extension.
getManager().activatePlugin(
      ext.getDeclaringPluginDescriptor().getId());
// Get plug-in class loader.
ClassLoaderclassLoader = getManager().getPluginClassLoader(
      ext.getDeclaringPluginDescriptor());
// Load Tool class.
Class toolCls = classLoader.loadClass(
      ext.getParameter("class").valueAsString());
// Create Tool instance.
tool = (Tool) toolCls.newInstance();
// Initialize class instance according to interface contract.
tool.init(toolComponent);
```

c) JSPF: Java Simple Plugin Framework[37]

Se trata de un proyecto del Centro de Investigación Alemán para la Inteligencia Artificial (DFKI) que fue construido pensando en proyectos de pequeña y mediana envergadura.

Los autores aconsejan su uso especialmente para la generación de prototipos rápidos, para desarrollos cuya implementación cambia con frecuencia y se desea tener un código limpio o para la reutilización de componentes en general.

Algunas de las ventajas que podemos encontrar acerca de este framework son:

- Se pueden cargar plugins de jars distintos al de la aplicación principal, de un directorio, de una ruta de clases e incluso a través de HTTP, aunque nos indican que no es seguro y que no es recomendable.
- Es multihilo.
- Incorpora typesafe (esto significa que el compilador validará los tipos durante la compilación y lanzará un error si se intenta asignar el tipo incorrecto a una variable).
- Gestiona la dependencia entre plugins.
- Puede configurarse con o sin ficheros XML.
- Soporta almacenamiento en caché.
- Proporciona plugins adicionales para exportar otros plugins a Javascript, JSON, LipeRMI, XMLRPC, Delicia XMLRPC o ERMI.

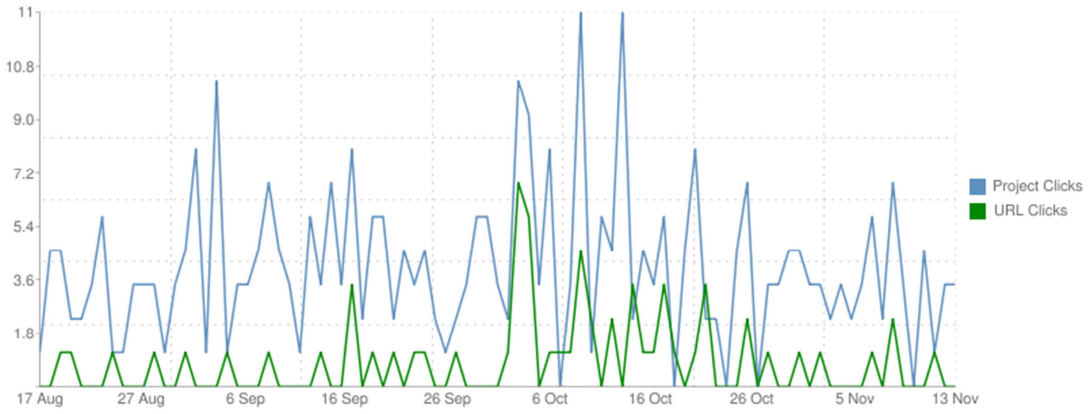

Figura 60. Estadísticas según freshmeat.net[38]

- Es multiplataforma, compatible con Windows, Mac, Linux e incluso con dispositivos con Android.
- Funciona con Applets, siempre y cuando éstos sean firmados y sin autodetección de classpath.

37 Página del autor de JSPF: http://code.google.com/p/jspf/
38 Página de examen de proyectos de código abierto:
http://freshmeat.net/projects/java-simple-plugin-framework/date_metrics

Desde la propia página del autor nos podemos descargar dos versiones de la plataforma: una con las librerías necesarias para incorporar el framework a nuestros desarrollos y otra con el código fuente.

El código fuente está dividido en dos partes, el núcleo del framework que se encuentra en la carpeta modules y unos ejemplos de cómo usar la plataforma en la carpeta test.

Siguiendo con el ejemplo para comparar las plataformas, a continuación se muestra cómo adaptaríamos nuestro editor simple de texto utilizando Java Simple Plugin Framework:

1. Generamos una interfaz que extienda a la interfaz del JSPF net.xeoh.plugins.base.Plugin en la que indicaremos las funciones que determinarán el comportamiento del tipo de plugins que la implementen. En nuestro caso con una función de inicialización nos vale:

```
importnet.xeoh.plugins.base.Plugin;

public interface CoolPlugin extends Plugin {
   publicvoidinicializate();
}
```

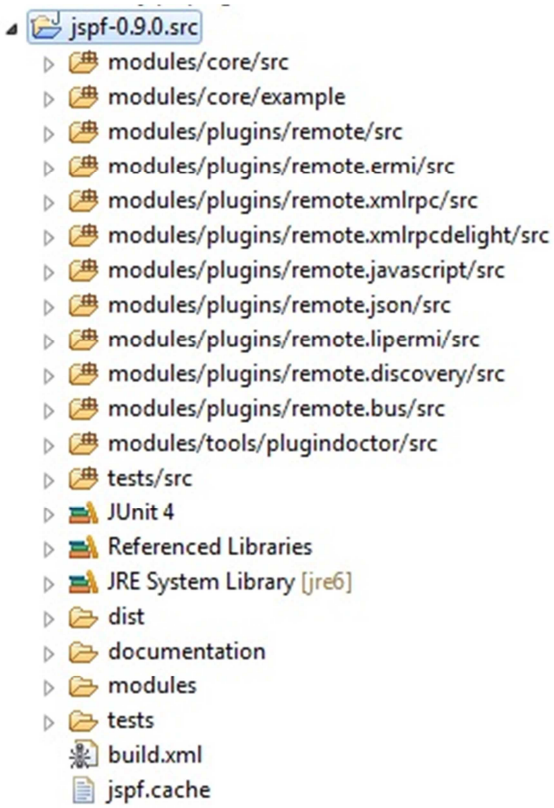

Figura 61. Estructura de módulos

2. Creamos nuestros plugins implementando la interfaz que acabamos de crear:

```
@PluginImplementation
public class EditorPlugin implements CoolPlugin{

   @Override
   public void inicializate() {
      construyeMenuEdicion();
   }

   public void construyeMenuEdicion() {
      JMenuItembuscar;
      Action accionCopiar;
      buscar = new JMenuItem(new AccionBuscar(Editor.areaTexto));
      accionCopiar = Editor.areaTexto.getActionMap().get(
            DefaultEditorKit.copyAction);
      accionCopiar.putValue(Action.NAME, "Copiar");

                  ...
```

3. En la aplicación principal creamos e inicializamos el gestor de plugins y cargamos e inicializamos los plugins de edición que acabamos de implementar:

```
public Editor()
{
      PluginManager pm;
      CoolPlugin plugin;

      this.inicializaComponentes();

      pm.addPluginsFrom(new ClassURI(EditorPlugin.class).toURI());
      plugin = pm.getPlugin(EditorPlugin.class);
      plugin.inicializate();

      v.setVisible(true);
}
```

Observaciones

- La única documentación que hemos encontrado es la del propio autor y no es muy compleja. En la descripción de la plataforma habla de muchas de las ventajas y funcionalidades y que ofrece pero no documenta la mayoría.
- El que sea compatible con la plataforma Android es un gran punto a su favor, ya que es parte de nuestro proyecto y lo tenemos que tener muy presente.
- Los “primeros pasos” con el framework son muy sencillos. Crear algo pequeño y fácil no cuesta nada ni entraña ninguna dificultad.
- El adaptar la aplicación al framework también es muy fácil, aunque hay que conocer la interfaz de los plugins que quieres inicializar.

- La creación de los plugins es muy sencilla, sólo hay que implementar una interfaz y añadir la información necesaria.

d) JIN-PLUGIN[39]

Jin-plugin se ofrece como una alternativa minimalista a otros frameworks de plugins en Java como OSGI o JPF.

Algunas características de este framework son:

- Además de ofrecer un sistema de gestión de plugins para Java, también tiene una parte para PHP.
- Soporta el registro de plugins como servicios, al igual que OSGI.
- Gestión de acciones. Las acciones son llamadas a métodos que se pueden hacer en diferentes sitios dentro de un plugin o un servicio.
- Tiene soporte para la actualización automática de los plugins.

El procedimiento básico de utilización de jin-plugin es el siguiente:

- La plataforma carga los plugins de una carpeta que se especifica en la aplicación padre. Por tanto, en la aplicación únicamente tendremos que especificar la ruta de la carpeta de plugins e inicializar el gestor de plugins.

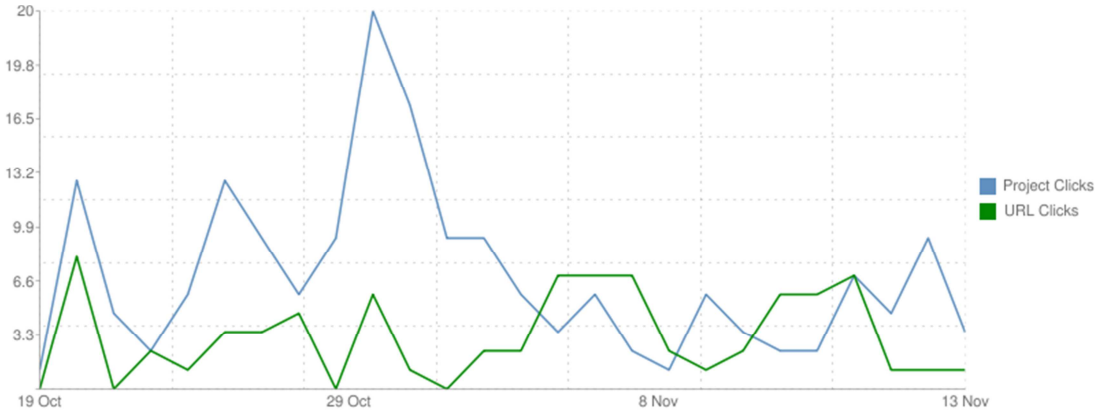

Figura 62. Estadísticas según freshmeat.net[40]

- Cada plugin consta al menos de una clase principal que implementa la interfaz com.estontorise.plugin.interfaces.Plugin, y un fichero denominado plugin.yamlque contiene la localización de la clase principal y las dependencias del plugin (si las tuviera). Dicho fichero debe estar dentro de la carpeta del plugin.

Cuando descargamos el código del framework lo primero que podemos apreciar es que el proyecto tiene totalmente diferenciada la parte Java de la parte PHP.

39 Página del autor de Jin-Plugin: http://code.google.com/p/jin-plugin/
40 Página de examen de proyectos de código abierto: http://freshmeat.net/projects/jin-plugin/date_metrics

Centrándonos en la parte Java, que es la que nos interesa, podemos observar que la estructura es muy sencilla y que se compone de 3 paquetes, uno de gestión de la plataforma, otro de interfaces que se deben implementar para su uso y un tercer paquete que contiene un ejemplo básico.

El procedimiento para introducir el frameworkjin-plugin en nuestra aplicación de prueba sería el siguiente:

1. En la aplicación, después de inicializar los componentes básicos del formulario (el frame, la barra de menú y la caja de texto) se inicializa el gestor de plugins:

```
public Editor() {
    this.inicializaComponentes();

    PluginManager pm;
    pm = PluginManagerFactory.createPluginManager("plugins");

    try {
        pm.init();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    v.setVisible(true);
}
```

Figura 63. Figura sobre la estructura de paquetes

2. Creamos el correspondiente plugin, tiene que haber unaclase principal que implemente el método com.estontorise.plugin.interfaces.Plugin. En nuestro caso, lo hemos dispuesto enun paquete llamado editionplugin y la clase principal es EditorPlugin.java.

```
public class EditionPlugin implements Plugin{

   @Override
   public void init(PluginManager pm) {
      construyeMenuEdicion();
   }

   public void construyeMenuEdicion() {
      JMenuItembuscar;
      Action accionCopiar;

      buscar = new JMenuItem(new AccionBuscar(Editor.areaTexto));

      accionCopiar = Editor.areaTexto.getActionMap().get(
            DefaultEditorKit.copyAction);
      accionCopiar.putValue(Action.NAME, "Copiar");
                  ...
```

3. Creamos un fichero plugin.yaml indicando cual es la clase principal del plugin (EditorPlugin.java) y si existen dependencias con otros plugins o no (en nuestro caso no).

4. Creamos una carpeta en la carpeta indicada en la aplicación padre (en nuestro caso plugins) y colocamos el fichero plugin.yaml.

```
plugin_class: editionplugin.EditionPlugin
```

Observaciones

- La única documentación que hemos encontrado es la del propio autor y prácticamente son pequeños ejemplos. Falta documentación para operaciones más complejas como el gestionar plugins que se encuentren en jars distintos al de la plataforma o si se puede utilizar en Android.
- Los "primeros pasos" con el framework son muy sencillos. Crear algo pequeño y fácil no cuesta nada ni entraña ninguna dificultad.
- El adaptar la aplicación al framework también es muy fácil, sólo hay que indicar la ruta del fichero de plugins e inicializar el gestor, del resto se encarga él.
- La creación de los plugins es muy sencilla, sólo hay que implementar una interfaz (cuyo nombre es muy intuitivo) y un método de inicialización.
- Creemos que la actualización remota de los plugins mediante una URL puede ser muy útil.
- No sabemos cómo de potente ni cuánto se puede complicar una plataforma tan simple.

Cuadro resumen de las plataformas

A continuación se muestra un cuadro resumen sobre cada una de las plataformas con las principales ventajas e inconvenientes de cada una de ellas.

PLATAFORMA	VENTAJAS	INCONVENIENTES
OSGI	• Aporta una gran funcionalidad.	• Muy compleja. • Requiere grandes cambios en la plataforma original.
JPF	• Arquitectura abierta. • Chequeo de integridad de plugins. • Chequeo de dependencias.	• Documentación escasa. • Problema de compatibilidad con eclipse.
JSPF	• Gran funcionalidad. • Compatible con Android.	• Documentación escasa.
JIN-PLUGIN	• La adaptación entre el framework y los plugins es muy sencillo. • Soporte para la actualización automática de los plugins. • La creación de plugins es muy sencilla.	• Documentación escasa.

Conclusiones

Una vez realizado el análisis de las plataformas y, en base a las ventajas e inconvenientes, se ha decidido optar por la plataforma de gestión de plugins Jin-Plugin.

Creemos que implementar el plugin en sí mismo tendría una complejidad similar en cada una de las plataformas, por lo que nuestro interés se centra, por tanto, en la conexión del plugin a la herramienta base. Consideramos, por tanto, que uno de los puntos más críticos estará en la conexión entre la plataforma y el gestor de plugins. Habiendo analizado cada una de ellas, llegamos a la conclusión de que Jin-Plugin es la que más sencilla hace esta tarea.

Por otra parte, aunque la herramienta OSGI aporta una gran documentación, cosa de la que Jin-Plugin carece, creemos que OSGI es demasiado compleja y que aporta una gran funcionalidad que no necesitamos realmente, por lo que preferimos usar una herramienta más sencilla y fácil de manejar.

Capítulo IV
Desarrollo del proyecto

Tabla de contenidos

4. Desarrollo del proyecto

4.1 Introducción

En este capítulo se va a proceder a la descripción de los distintos patrones de diseño así como de las tecnologías que se implementan en el proyecto. De este modo, se realizará un estudio sobre su estructura, que aportan y cuáles son las principales motivaciones de uso de los mismos. También se va a proceder a la explicación de los hitos que nos planteamos al inicio del proyecto.

Se explicará el patrón Modelo Vista Controlador, que es una metodología básica de arquitectura de software que separa los datos de una aplicación, la lógica de negocio y la interfaz de usuario. Es ampliamente utilizada por diferentes motivos que se comentarán posteriormente.

Tras esto, se definirá más concretamente qué son y por qué se usan Google Guice, que es, básicamente, un framework para inyección de dependencias[41] para la plataforma Java y JUnit, un framework para escribir test reutilizables.

4.2 Hitos del proyecto

Lo primero de todo es establecer los hitos del proyecto. Concretamente, planteamos que fueran cuatro, a saber:

- Hito 1: Análisis de la plataforma y captura de requisitos.
- Hito 2: Adaptación inicial del motor de juegos para facilitar el desarrollo del sistema de extensiones.
- Hito 3: Desarrollo del sistema de extensiones para la plataforma.
- Hito 4: Desarrollo de varias extensiones de ejemplo.

Inicialmente, nos planteamos realizar un análisis de la plataforma. Para la consecución de este primer hito, realizamos un estudio sobre el código de e-Adventure. Además, implementamos un juego educativo sobre medicina

41 Inyección de dependencias http://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias

para observar las ventajas y desventajas que la plataforma básica tenía y así poder comprobar que funciones eran difíciles de obtener.

Una vez hecho esto, y elegido Jin-Plugin como sistema base de gestor de plugins, hicimos la adaptación necesaria de e-Adventure para poder acoplar el framework de extensiones, cumpliendo así el segundo hito de la lista.

A continuación, implementamos el sistema de extensiones. Este ha sido basado en Jin-Plugin, si bien es cierto que tuvimos que realizar una serie de cambios para poder obtener toda la funcionalidad que considerábamos necesaria.

Por último, hemos desarrollado una serie de plugins de ejemplo, que son explicados más en detalle posteriormente, concretamente en el capítulo VI. Basicamente, hemos implementado videojuegos educativos siguiendo la filosofía de e-Adventure.

4.3 Patrón Modelo Vista Controlador

El patrón Modelo Vista Controlador[42] es un estilo de arquitectura de software que separa los datos de una aplicación, la lógica de negocio y la interfaz de usuario. Este patrón implica un diseño que desacople estas tres capas con la finalidad de mejorar la reusabilidad y que de esta forma las modificaciones en la vista (que suele ser lo que cambia con mayor frecuencia) impacten en menor medida en la lógica de negocio, en los datos o en la interfaz de usuario.

Para este proyecto, se quería, ante todo, una fácil estructuración del código y que a la vez, este fuera flexible. Antes estos objetivos básicos, se cree conveniente usar la metodología de Modelo-Vista-Controlador, al facilitar el uso de estos aspectos requeridos entre sus ventajas. También se desarrolla puesto que permite una separación real de los tres apartados que forman parte de una aplicación (los datos, la implementación del sistema y la interfaz de usuario).

En el caso concreto del proyecto, podemos observar el uso de este patrón a la hora de generar cada uno de los plugins. Así, cargaremos los datos desde ficheros .xml a través del interfaz que ofrece el framework Jin-Plugin. La implementación en sí del sistema, las diferentes funcionalidades que ofrece el plugin, y la interfaz de usuario también se usarán de manera independiente, pudiendo variar cualquiera de ellas sin necesidad de generar un cambio obligatorio en la otra.

Creemos, además, que al tratarse de un proyecto englobado en uno mucho mayor, como es E-Adventure, este debe poder ser fácilmente mantenido. Necesitamos tener una herramienta flexible para que, en el caso de necesitarse/desearse en un futuro, pueda ser modificado de manera sencilla.

---

42 Breves reseñas del patrón Modelo-Vista-Controlador:
http://www.proactiva-calidad.com/java/patrones/mvc.html
http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

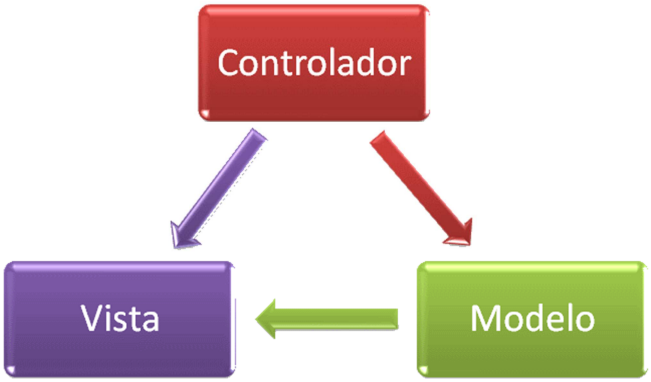

Figura 64. Relación entre las capas de MVC[43]

El flujo de trabajo que se sigue con MVC es el siguiente:

1) El usuario interactúa con la interfaz de algún modo: pulsando un botón, un enlace, etc.
2) El controlador recibe la notificación de la acción (mediante los objetos de la vista-interfaz, como por ejemplo por un evento)
3) El controlador accede al modelo y lo actualiza si es necesario conforme a lo establecido en la lógica de negocio para esa acción en concreto.
4) El controlador delega a los objetos de la vista la tarea de generar la nueva interfaz de usuario actualizada. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario, pero el modelo no debe tener conocimiento sobre la vista, son independientes.
5) La interfaz de usuario se queda a la espera de nuevas interacciones y cuando se produce alguna se vuelve a repetir el ciclo.

Esta arquitectura permite el uso de otros muchos patrones de diseño, los cuales son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un ejemplo puede ser el patrón Command[44], que suele ser utilizado por controladores complejos y que permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación ni el receptor real de la misma, o el patrón Observador[45], que define una dependencia del tipo uno a muchos entre objetos, de manera que cuando un objeto cambia su estado el observador se encarga de notificar ese cambio a todos los dependientes, y que puede ser muy útil a la hora de establecer la relación entre el modelo y la vista.

A continuación vamos a poner un pequeño ejemplo[46] de cómo aplicaríamos el Modelo Vista Controlador a una calculadora sencilla que convierte de euros a pesetas.

Lo primero que debemos hacer es pensar qué funcionalidad debería ir en cada capa. En este caso lo hemos distribuido de la siguiente forma:

43 Figura del Modelo-Vista-Controlador http://www.boliviaonrails.com/2008/07/17/que-es-ruby-on-rails
44 Patrón "Command" http://es.wikipedia.org/wiki/Command_(patr%C3%B3n_de_dise%C3%B1o
45 Patrón "Observer" http://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o)
46 Ejemplo de conversor de pesetas a euros siguiendo el patrón Modelo-Vista-Controlador http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf

- En el Modelo irá la parte correspondiente a las operaciones de datos, es decir, estableceremos la cifra del cambio (166.386) y realizaremos la operación de conversión.
- En el Controlador nos encargaremos de la captura de los eventos de la interfaz y de ejecutar la acción correcta en función del evento recibido.
- En la vista nos encargaremos tanto de la captura de datos introducidos por el usuario como de mostrar el resultado de la operación solicitada.

Este diseño de la calculadora tiene múltiples ventajas frente al diseño tradicional, por ejemplo:

- El modelo puede ser reutilizable a múltiples calculadoras.
- La vista puede modificarse tanto como se quiera, no afectaría a la funcionalidad de la calculadora. Incluso se podría encargar los cambios de la interfaz a alguien que no supiera qué hace o cómo funciona exactamente la calculadora.
- Con respecto a la lógica de negocio ocurre de manera análoga a las otras capas. Si se quisiera cambiar la funcionalidad de la calculadora podría hacerse de manera muy natural y sencilla. Por ejemplo, podríamos querer incrementar la funcionalidad de la calculadora para añadir también la conversión a libras esterlinas. Mediante la anexión del código necesario y un pequeño cambio en la vista, podríamos disponer fácilmente de esta nueva funcionalidad deseada.

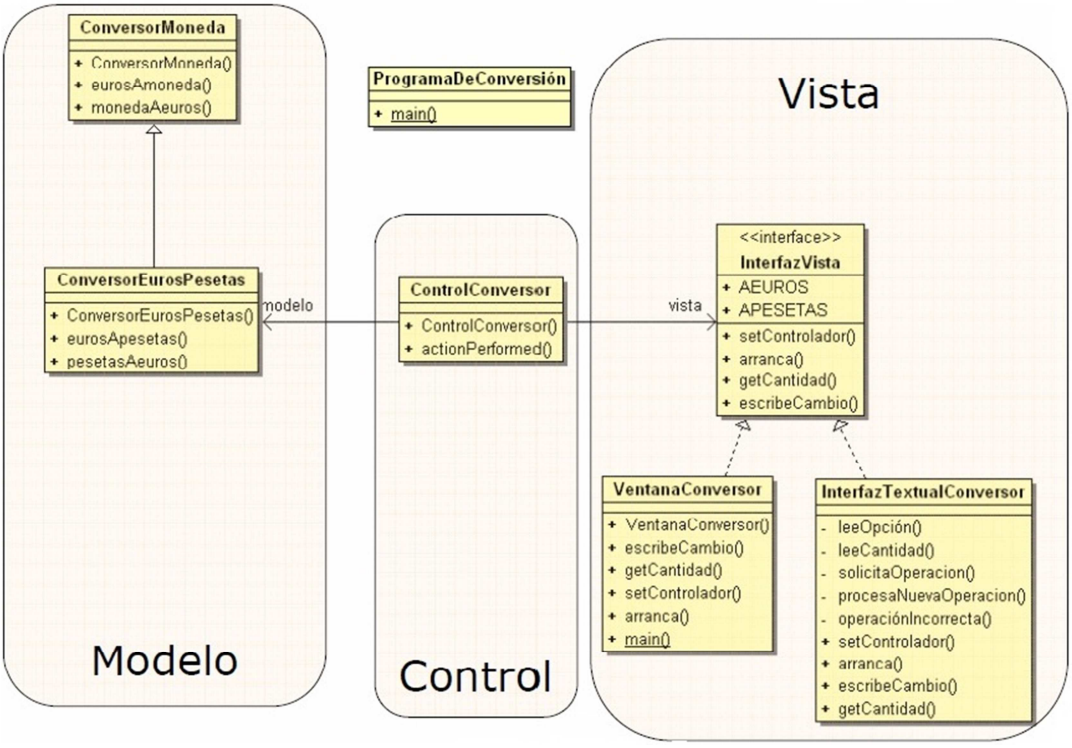

Figura 65. Modelo-Vista-Controlador para el ejemplo del conversor

4.4 Google Guice

Google Guice [47] es un framework para inyección de dependencias para la plataforma Java publicado por Google y bajo licencia Apache. Para comprender mejor el concepto de inyección de dependencias vamos a comenzar hablando de qué es.

La inyección de dependencias es un patrón de diseño cuyo principio fundamental es separar el comportamiento de la resolución de dependencias[48], que son las aplicaciones o bibliotecas software que son necesarias para la correcta ejecución del programa.

Cuando queremos afrontar el desarrollo de aplicaciones potencialmente grandes, donde podemos tener distintos equipos de desarrolladores codificando distintos módulos para una misma aplicación, nos encontramos con la necesidad de realizar tests automatizados de una o más clases, o dependencias con código de otro módulo que ni siquiera se encuentra desarrollado. En ese caso es necesaria la inyección de dependencias.

La inyección de dependencias es un patrón de diseño orientado a objetos en el que se suministran objetos a una clase en lugar de ser la propia clase quien lo cree. Por ejemplo, si tuviéramos una clase como esta:

```
public class Vehiculo {

  private Motor motor = new Motor();

  public Double enAceleracionDePedal(int presionDePedal) {
    motor.setPresionDePedal(presionDePedal);
    int torque = motor.getTorque();
    Double velocidad = ... //realiza el cálculo
    return velocidad;
  }

}
```

Una forma de aplicar el patrón de inyección de dependencias sería delegar la creación de la clase motor a otra clase distinta, así abstraeríamos el vehículo del motor, de la siguiente forma:

```
public class Vehiculo {

  private Motor motor = null;

  public setMotor(Motor motor){
    this.motor = motor;
```

47 Proyecto Google Guice http://code.google.com/p/google-guice/
Introducción a Google Guice http://www.javabeat.net/articles/29-introduction-to-google-guice-1.html
Breve reseña de Google Guice http://juancavallotti.blogspot.com/2010/01/google-guice-breve-resena.html
48 Dependencias de software http://es.wikipedia.org/wiki/Dependencias_de_software

```
  }

  public Double enAceleracionDePedal(int presionDePedal) {
    Double velocidad = null;
    if (null != motor){
      motor.setPresionDePedal(presionDePedal);
      int torque = motor.getTorque();
      velocidad = ... //realiza el cálculo
    }
    return velocidad;
  }

}

public class VehiculoFactory {

  public Vehiculo construyeVehiculo() {
     Vehiculo vehiculo = new Vehiculo();
     Motor motor = new Motor();
     vehiculo.setMotor(motor);
     return vehiculo;
  }

}
```

La forma habitual de implementar este patrón en Java es mediante un contenedor y objetos POJO (Plain Old Java Object, siglas utilizadas para enfatizar el uso de clases simples y que no dependen de un framework en especial).

El contenedor suele ser implementado por un framework externo a la aplicación, y aquí es donde entra Google Guice.

Google Guice implementa el estándar JSR 330 para la inyección de dependencias, que es un estándar creado por Google y SpringSource[49] que especifica la forma de obtención de objetos de tal manera que se maximice la reutilización, la capacidad de prueba y mantenimiento en comparación con los enfoques tradicionales, como constructores, fábricas y los localizadores de servicios (por ejemplo JNDI).

Posee algunas características, como por ejemplo:

- Inyección de dependencias en constructor, utilizando setters y directamente en los atributos de la clase.
- Intercepción de métodos con filtro por anotaciones (especialmente útil en la Programación Orientada a Aspectos[50]).
- Configuración programática de los módulos de la aplicación.

49 Spring Source http://www.springsource.org/
50 Programación Orientada a Aspectos
http://es.wikipedia.org/wiki/Programaci%C3%B3n_Orientada_a_Aspectos

- Definición de fábricas o proveedores y ámbitos para las instancias inyectadas.

Y algunos de los beneficios de utilizar este framework son:

- Generalización de aspectos en las aplicaciones y su consiguiente reducción de errores.
- El código es mantenido de manera más sencilla, testeable y mucho más reutilizable.

En nuestro proyecto es indispensable el uso de Google Guice puesto que E-Adventure utiliza ampliamente este framework y pensamos que es una filosofía de programación novedosa pero con un gran potencial de trabajo.

Además, como e-Adventure está pensado para múltiples plataformas, es una buena práctica para abstraer el motor de la plataforma en concreto

A continuación, vamos a mostrar un ejemplo muy sencillo de cómo utilizaríamos Google Guice en un gestor de cumpleaños.

```
@Singleton
public class BirthDateManager {

	private EntityManager em;

	@Inject
	public BirthDateManager(EntityManager em) {
		this.em = em;
	}
}
```

- Con la anotación @Singleton le estamos indicando al inyector que esta clase implementa el patrón de diseño con ese mismo nombre[51].
- Con la anotación @Inject le indicamos al inyector que debe inyectar dependencias sobre este método.

Ahora observamos la clase que define el módulo donde se configurará el inyector:

```
public class BirthDateModule extends AbstractModule {

	@Override
	protected void configure() {
		bind(EntityManager.class).toProvider(PersistenceFacade.class);
	}

}
```

51 Singleton http://es.wikipedia.org/wiki/Singleton

Con este código le indicamos al inyector que debe obtener la dependencia de un proveedor, y que en este caso el proveedor va a ser la fachada de persistencia, cuyo código mostramos a continuación:

```
public class PersistenceFacade implements Provider<EntityManager> {

        @Override
        public EntityManager get() {
                return emf.createEntityManager();
        } ...
}
```

Para que Guice sepa cuál es nuestra clase proveedora deberá implementar la interfaz Provider. Finalmente debemos añadir el código que une todo y ya tendríamos Google Guice funcionando en nuestra aplicación:

```
Injector in = Guice.createInjector(new BirthDateModule());
BirthDateManager bdm = in.getInstance(BirthDateManager.class);
```

4.5 JUnit

JUnit[52] es un framework simple para escribir test reutilizables. Se trata de una instancia de la arquitectura xUnit de testeo de frameworks de pruebas unitarias.

JUnit permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

- Caso más simple de pruebas:

La manera más sencilla de prueba es como expresión en el “debugger”. Se pueden depurar expresiones sin necesidad de recompilar y, por tanto, se puede esperar hasta ver el resultado de la ejecución para cambiar el código que necesitemos. También se pueden escribir expresiones de prueba como declaraciones que imprimen a la salida estándar.

Ambas maneras de crear los tests son limitadas puesto que requieren de una persona que analice los resultados obtenidos. Por esto, sólo puede ejecutarse una expresión de depuración cada vez y un programa con demasiadas operaciones de imprimir resultados puede provocar confusión.

52 Página oficial de JUnit http://www.junit.org/
Breve reseña http://es.wikipedia.org/wiki/JUnit
Breve ejemplo de uso de JUnit http://junit.sourceforge.net/doc/cookbook/cookbook.htm
Código de ejemplo de un test JUnit http://www.java2s.com/Open-Source/Java-Document/Testing/junit/org/junit/samples/SimpleTest.java.htm

Los test de JUnit no requieren de una persona para interpretar o juzgar los resultados, por lo que es mucho más sencilla la ejecución de muchas pruebas simultáneamente.

El ejecutar tests que comprueben el correcto funcionamiento de cada una de las partes del framework de plugins es algo básico, como en cualquier aplicación. De hecho, consideramos que es una parte muy importante y que muchas veces este trabajo está infravalorado.

A continuación se muestran los pasos a seguir para realizar los tests:

1) Anotamos el método que queremos chequear con:

```
@org.junit.Test
```

2) Cuando queremos comprobar un valor, importamos org.junit.Assert.* estáticamente, llamamos a la función assertTrue() y pasamos una variable booleana que adquirirá el valor verdadero si el test se realizó correctamente.

Como ejemplo, para probar que la suma de dos cantidades de dinero (Money) con la misma moneda contiene un valor que es la suma de dos valores de Money, escribimos:

```
@Test public void simpleAdd() {
   Money m12CHF= new Money(12, "CHF");
   Money m14CHF= new Money(14, "CHF");
   Money expected= new Money(26, "CHF");
   Money result= m12CHF.add(m14CHF);
   assertTrue(expected.equals(result));
}
```

- Fixture(Accesorio)

Fixture se utiliza cuando tenemos dos o más test que operan de una forma similar sobre un conjunto de objetos.

El conjunto de objetos se denomina “test fixture”. Cuando se escriben tests, se puede comprobar que se emplea mucho más tiempo escribiendo código que en ejecutar los tests en sí.

Hasta cierto punto, se puede escribir el código de los accesorios (fixtures) más fácilmente poniendo especial atención en los constructores que se crean. Sin embargo, se puede ahorrar mucho más código compartiendo el código de diferentes fixtures.

Una gran mayoría de veces podrán ser reutilizados en bastantes tests

un mismo “fixture”. En cada caso se enviarán mensajes o parámetros ligeramente diferentes al “fixture”, que comprobará los diferentes resultados.

Cuando se dispone de un fixture común, se deben seguir una serie de pasos.

1) Añadir un campo por cada parte del “fixture”.
2) Anotar un método con @org.junit.Before e inicializar las variables en ese método.
3) Anotar un método con @org.junit.After para liberar cualquier recurso permanente que hayamos asignado al iniciar.

Por ejemplo, para escribir bastantes casos de pruebas que quieran trabajar con diferentes combinaciones de 12 francos suizos, 14 francos suizos y 28 dólares, primero creamos un “fixture”:

```
public class MoneyTest {
   private Money f12CHF;
   private Money f14CHF;
   private Money f28USD;

   @Before
  public void setUp() {
     f12CHF= new Money(12, "CHF");
     f14CHF= new Money(14, "CHF");
     f28USD= new Money(28, "USD");
   }
}
```

Una vez que tenemos el “fixture”, podemos escribir tantos casos de pruebas como se deseen. Añadiendo tantos métodos de pruebas (anotados con @Test) como se quieran.

- Excepciones:

Verificar que las excepciones que lanza el código son las esperadas y así comprobar que el código se comporta como se espera de él en situaciones excepcionales es una parte muy importante de la programación.

Por ejemplo, tenemos la siguiente situación:

```
new ArrayList<Object>().get(0);
```

Este código debería lanzar una excepción del tipo IndexOutOfBoundsException.

La anotación @Test permite incluir un parámetro opcional “expected” que toma como valores subclases de la superclase “Throwable”. Si queremos verificar que ArrayList lanza la excepción correcta, se debe escribir:

```
@Test(expected= IndexOutOfBoundsException.class)
```

```
public void empty() {
   new ArrayList<Object>().get(0);
}
```

- Método para ejecutar las pruebas creadas y recolectar los resultados:

Lo primero que se ha de hacer es implementar los test que deseemos ejecutar.

JUnit provee herramientas para definir la serie de tests que queremos ejecutar y mostrar los resultados. Para ejecutar estos tests y ver los resultados en consola debemos ejecutar la siguiente instrucción en un programa Java:

```
org.junit.runner.JUnitCore.runClasses(TestClass1.class, ...);
```

Otra opción es ejecutarlo desde línea de comandos, con la clase del test y el JUnit incluido en el classpath o ruta de clases.

```
java org.junit.runner.JUnitCore TestClass1 [...other test classes...]
```

Se pueden hacer accesibles las clases de los test de JUnit 4 a una serie de pruebas diseñada para trabajar con versiones anteriores de JUnit declarando un método estático “suite” que devuelve un test.

```
public static junit.framework.Test suite() {
   return new JUnit4TestAdapter(Example.class);
}
```

Capítulo V
Descripción del framework

Tabla de contenidos

5. Descripción del framework de plugins

5.1 Introducción

A continuación se va a proceder a la explicación completa del framework de plugins implementado y que constituye una de las partes más importantes del proyecto.

5.2 Framework en el estado inicial

Como aparece en el capítulo III de la memoria, se decidió implementar el sistema de plugins en base al framework de plugins Jin-Plugin.

En principio, la conexión entre la herramienta de gestión de plugins y la plataforma e-Adventure se trata de una tarea sencilla. A continuación se detallan las acciones realizadas para incluir los plugins en e-Adventure:

- E-Adventure lee, mediante un gestor de plugins o plugin manager, los plugins que se deben cargar, que serán aquellos que se especifiquen en un fichero *plugin.yaml*, y que deben estar en una carpeta *plugins* especificada. Hemos creído conveniente a tal efecto disponer de una carpeta en la que cualquiera que desee un nuevo plugin debe introducirlo.
- El fichero *plugin.yaml* contiene la localización de la clase principal y las dependencias del plugin (si las tuviera).
- El plugin manager inicializa automáticamente los plugins y, tras esto, ya los tendremos cargados en la aplicación y podemos utilizarlos.

5.3 Modificaciones al framework

En este apartado se hace un seguimiento de la evolución que ha ido siguiendo el framework de Jin-Plugin con respecto a las dificultades de

implementación y las soluciones que se han ido adoptando según avanzaba el proyecto.

5.3.1 Plugin del puzle

Uno de los problemas encontrados al ponernos a implementar el plugin manager fue que, si bien es cierto que al iniciar el editor se cargaban los plugins, no siempre querremos que estos se incluyan en el editor de e-Adventure o que lo hagan de la misma forma. En algún caso querremos que se carguen en el panel derecho de la interfaz, otros en el panel superior y/o que no aparezcan.

Esto implica que deberían existir diferentes maneras de inicializar cada uno de los plugins. Pero Jin-Plugin sólo dispone de un método común *init* que ejecuta la inicialización (usando el método *init*) de cada uno de los plugins cargados. Nos dimos cuenta de que necesitaríamos extender la funcionalidad de Jin-Plugin para disponer de una lista de plugins que ejecutaran los distintos métodos de cada plugin según las necesidades.

Las clases proporcionadas por el framework resultaban demasiado básicas, por lo que tuvimos que modificar un poco la estructura del framework en sí.

Estas son las clases principales que nos han surgido:

- La clase EAdPluginManager, que extiende PluginManager.

  Esta extensión se realizó por que queríamos disponder de una lista de plugins para manipular cada uno de ellos de manera individual, la ruta a los plugins almacenada y la lista de los plugins cargados.

- La interfaz EAdPlugin, que extiende Plugin.

  Necesario puesto que queríamos que el plugin almacenara un nombre que funcionara como identificador. También es necesario porque según nuestra estructura de plugin este estará dividido en dos partes: La parte del motor, de tipo EAdEnginePlugin, y la parte del editor, de tipo EAdEditorPlugin.

- EAdEnginePlugin

  Se trata de una interfaz que deberá implementar obligatoriamente cada uno de los plugins. Contiene los métodos necesarios para todos los plugins en la parte del motor.

1) Un método *render* para el dibujado del plugin una vez en ejecución.

- EAdEditorPlugin

Es una interfaz que define tres métodos que consideramos básicos para un plugin en la parte del editor:

1) getLeftPanelButton() de tipo EAdButton, para generar un botón en el panel izquierdo del editor de EAdventure.
2) getTopPanelButton() de tipo EAdButton y, que de manera análoga al izquierdo, generará un botón en el panel superior.
3) getPanel() de tipo EAdCanvasPanel para generar el Canvas Panel del plugin en el editor.

5.3.2 Plugin del puzle arrastrando piezas

Para la implementación de este plugin no se ha incrementado la funcionalidad del gestor de plugins puesto que, con los cambios realizados en el plugin anterior, ésta nos es suficiente.

Lo que se debe comentar sobre todo sobre el framework en base a nuestro uso de él en este plugin es el uso de dependencias, ya que, usaremos las clases que forman parte del editor del primer plugin para este segundo.

Para realizar la dependencia entre plugins y como habíamos previsto al elegir este framework, tan sólo debemos incluir la ruta al plugin del que se depende en el archivo *plugin.yaml* bajo la etiqueta *dependencies* de la siguiente manera:

```
dependencies: EAdPuzzle
```

ya que EAdPuzzle es el nombre de nuestro primer plugin.

El manager de plugins, al iniciar cada uno de estos, inicializa también cada uno de los plugins de los que depende. Si alguno de estos plugins de los que depende no puede ser inicializado, ya que no se encuentra en la carpeta con el resto de plugins, el plugin original no puede hacerlo tampoco, generándose un error.

5.3.3 Resto de plugins

Con respecto al resto de plugins no hemos tenido que retocar más el framework puesto que ya cumple todas las funcionalidades requeridas.

Capítulo VI
Casos de estudio

Tabla de contenido

6. Introducción

Este capítulo se encuentra dividido en dos partes. En la primera trataremos nuestras primeras impresiones a la hora de manejar la herramienta de e-Adventure 1.x; errores encontrados, así como dificultades y ventajas a la hora de utilizarlo a nivel usuario.

En la segunda parte se explicarán los plugins que se han desarrollado para la herramienta, ya que estos ejemplos son los que han ido aportando solidez al framework desarrollado para la inclusión de plugins y los que han hecho aflorar los problemas y dificultades de la integración de los plugins con el núcleo.

6.1 Primeros pasos con e-Adventure

Para aprender la funcionalidad y características de la aplicación e-Adventure, se ha desarrollado con ella un sencillo juego como primera toma de contacto. Esta tarea ha sido incluida entre las primeras del PFC, con el fin de establecer una base que nos diera a conocer la herramienta desde la perspectiva del usuario. De esta forma, hemos podido ver objetivamente el tipo de funciones que se echan en falta, así como posibles mejoras que se puedan aplicar; es decir, la realización del juego nos ha ayudado a analizar e-Adventure para conocer su funcionamiento y deducir extensiones que incorporarle, mejorando la funcionalidad existente o ampliándola.

El juego que hemos hecho para familiarizarnos con e-Adventure tiene como temática los primeros auxilios. Se trata de un juego en tercera persona que plantea tres posibles situaciones de emergencia, en las que el jugador tiene que ser capaz de solucionar el problema. Estas situaciones varían entre el atragantamiento, dolor torácico y quedarse inconsciente. Toda la documentación y especificación sobre este juego nos ha sido facilitada por los tutores, puesto que se tenía pensado llevar a cabo independientemente del proyecto que nos ocupa; por lo tanto nuestra única tarea ha sido elaborarlo

con e-Adventure, pudiéndonos centrar en los aspectos de análisis y aprendizaje de la herramienta.

Las primeras impresiones positivas sobre e-Adventure tras comenzar a crear el juego son:

- Una interfaz gráfica sencilla y pragmática: Se muestran botones grandes y secciones bien diferenciadas para cada funcionalidad, lo que facilita al usuario aprender más rápidamente su uso e interacción. Está pensado para que cualquier persona sin muchos conocimientos informáticos pueda utilizarla con poco esfuerzo, y poco tiempo de aprendizaje.
- Cada sección cuenta con una funcionalidad muy concreta para desarrollar los juegos, por lo que uno pronto se hace la idea de cómo se utilizan, y la forma en que se van creando con su uso.
- Extensa documentación, así como tutoriales que enseñan de forma sencilla a utilizar la aplicación.

Al ir avanzando en la realización del juego comprobamos la facilidad con la que se pueden desarrollar este tipo de juegos 'point and click' gracias a e-Adventure, que ofrece un gran número de herramientas para incluir muchas de las características que tienen estos juegos. Igualmente, nos encontramos con dificultades dependiendo de la tarea que queríamos hacer. Por ejemplo, echamos en falta una extensión sencilla de edición de imágenes, que nos permitiera redimensionar, recortar, etc..., algunos dibujos para el juego, sin necesidad de tener que emplear otros programas de edición fuera de e-adventure. Nos dio la sensación, de que esta prestación aportaría muy positivamente en su funcionalidad. También nos pareció poco práctica la gestión de conversaciones y diálogos, sobre todo para un juego en el que este aspecto es fundamental, como es el caso del de primeros auxilios, en el cual son muy utilizados. Cuando se trata de una conversación pequeña y simple no hay problema, pero si comienza a extenderse con muchos nodos y caminos, llega a complicarse su tratamiento.

Otra característica que nos pareció muy positiva es la inclusión de variables y flags, que aportan más dinamismo y múltiples opciones de interacción a los juegos, y su utilización es muy sencilla.

Una de las funciones de las que más se nota su falta es la existencia de plantillas a la hora de relacionar conceptos. En el juego de ejemplo que hemos llevado a cabo, se necesita reiteradamente el típico caso de selección entre múltiples opciones, actividad que se complica un poco sin contar con un esquema más orientado a ello. Presenta más importancia teniendo en cuenta que en juegos educativos este tipo de ejercicios son muy empleados. Para poder hacerlo hemos necesitado incluir cuatro imágenes, que hemos tratado con un programa de edición exterior, y controlar el comportamiento de cada imagen por separado dependiendo de la elección del jugador. Queda claro, que contando con una extensión orientada a esta acción se podría realizar mucho más fácilmente.

A continuación, se muestra una lista de las impresiones tras la elaboración del juego y de extensiones con las que podría contar e-Adventure para solventar los temas expuestos, entre otros. Comentar que, actualmente, muchas de estas ideas ya han sido mejoradas o solucionadas.

- Nos planteamos la opción de no establecer imagen de fondo para una escena, y poder elegir, por ejemplo un color como fondo de escenario. Pensamos que esto sería muy útil en pantallas de opciones, o de unir conceptos con flechas. Además encontramos varios problemas a la hora de establecer la imagen de fondo, como puede ser que el tamaño o la resolución no sean configurables con el editor.
- Pensamos que sería útil el que se pudiera lanzar un efecto en una escena sin necesidad de interacción por parte del jugador. Un ejemplo que se nos ocurre es que al avanzar una escena se pudiera lanzar un evento de aleatoriedad.
- Encontramos algunos problemas de rendimiento a la hora de cargar muchas imágenes y vídeos.
- No se ofrece mucho soporte para tareas como pruebas de elegir conceptos, o relacionarlos mediante flechas, por ejemplo, cosas de este tipo.
- Los botones de añadir y eliminar efectos, salidas etc. nos parecen demasiado pequeños.
- Configurar el texto de las conversaciones de manera independiente para cada ítem de conversación.
- Al introducir texto en una imagen no se pueden establecer las coordenadas por teclado, únicamente se puede con la flechas de dirección del teclado, lo cual hace un poco tediosa esta tarea.
- La accesibilidad del programa no es muy buena para los netbooks. El programa aparece con la parte superior por encima del límite superior de la pantalla, por lo que no es posible redimensionarlo o moverlo. También se hace difícil trabajar con el panel de la izquierda, ya que al elegir una opción no se visualiza más que una de ellas (y no entera).
- Los efectos de probabilidad son solo entre 2 efectos, y solo se puede lanzar 1 efecto por vez. Creemos que sería de gran utilidad que esta parte del editor fuera muy configurable, poder tener varios rangos de probabilidad con varios efectos a lanzar como consecuencia y así dar un toque de aleatoriedad a los videojuegos y que estos sean más interesantes y variables. Otra funcionalidad asociada a esto que se nos ocurre es que pudiéramos asignar un valor a una variable o un flag en función de una probabilidad.
- También nos encontramos con que, tras llevar gran parte del videojuego creada y por tanto mucha información almacenada sobre las escenas o las imágenes cargadas, el fichero base del videojuego se corrompía y nos impedía seguir avanzando.

6.2 Desarrollo de plugins

6.2.1 Plugin del puzle

Alcance

El plugin, al igual que la mayoría de los plugins desarrollados en el proyecto, se compone de dos partes diferenciadas pero necesarias para el funcionamiento del mismo.

En la parte del editor se compondrá de una pestaña que muestra un panel donde se pueden configurar los principales aspectos de configuración del plugin, como puede ser la imagen, complejidad o texto introductorio al plugin.

En la parte del motor, se han de desarrollar los algoritmos necesarios para poder desplazar las piezas al hueco y poder llegar a la imagen original resolviendo así el plugin. El usuario se encontrará con un puzle que, al presionar un botón se desordenarán las piezas. Para moverlas deberá hacer clic con el ratón en una de las piezas que se encuentre en una posición correcta de movimiento (arriba, abajo, a la izquierda o a la derecha del hueco) y después en la pieza del hueco y acto seguido se intercambiarán las piezas del hueco y de la casilla seleccionada.

Requisitos

Esta extensión debe permitir generar puzles a partir de una imagen dada. En la ejecución del juego la imagen se dividirá en diferentes porciones que se desordenarán y el usuario deberá volverlas a ordenar siguiendo las reglas del juego del 8Puzzle, que son:

- Se da un tablero con N casillas, de las cuales una casilla se encuentra vacía.
- Dicha casilla vacía, con movimientos horizontales, verticales, hacia la izquierda o derecha, debe ser desplazada e intercambiada con alguno de sus vecinos, de manera que, dada una configuración inicial se llegue a una configuración final (meta).

En la plataforma, para generar los puzles se cuenta con:

- Panel que se adhiera a la interfaz gráfica principal y que contenga campos de texto para introducir datos representativos para cada rompecabezas, como pueden ser su nombre, descripción, ruta donde se encuentra la imagen, dificultad, etc...
- Botón para seleccionar la imagen del puzle mediante una ventana de diálogo que permita elegir el archivo.
- Selección de distintos niveles de dificultad para indicar la división que haya que hacer de filas por columnas para la imagen.
- Casilla de verificación para indicar si se da la opción de rendirse, y posibilidad de configurar un tiempo determinado de juego.

Respecto a la interacción del usuario cuando se está ejecutando este sistema:

- Visualización de la imagen dividida y botón para desordenarlo de forma aleatoria.
- Desplazamiento de las casillas utilizando el ratón.
- Finalización del juego cuando el usuario consiga formar el puzle correctamente (llegando al dibujo original), tras alcanzar un tiempo límite de juego o cuando el usuario decide rendirse.

Diseño del plugin

Entre las ideas generales en el diseño de este plugin podemos distinguir las siguientes:

- El número de piezas posibles para dividir la imagen, relativo al grado de dificultad, se establece en tres configuraciones distintas:
  - Nivel fácil: división en 3 x 3
  - Nivel medio: división en 4 x 4
  - Nivel difícil: división en 5 x 5

El editor se va a caracterizar por disponer de una interfaz gráfica con las características descritas en la sección de requisitos. Se facilita un prototipo que muestra el diseño de dicha interfaz (Figura 66).

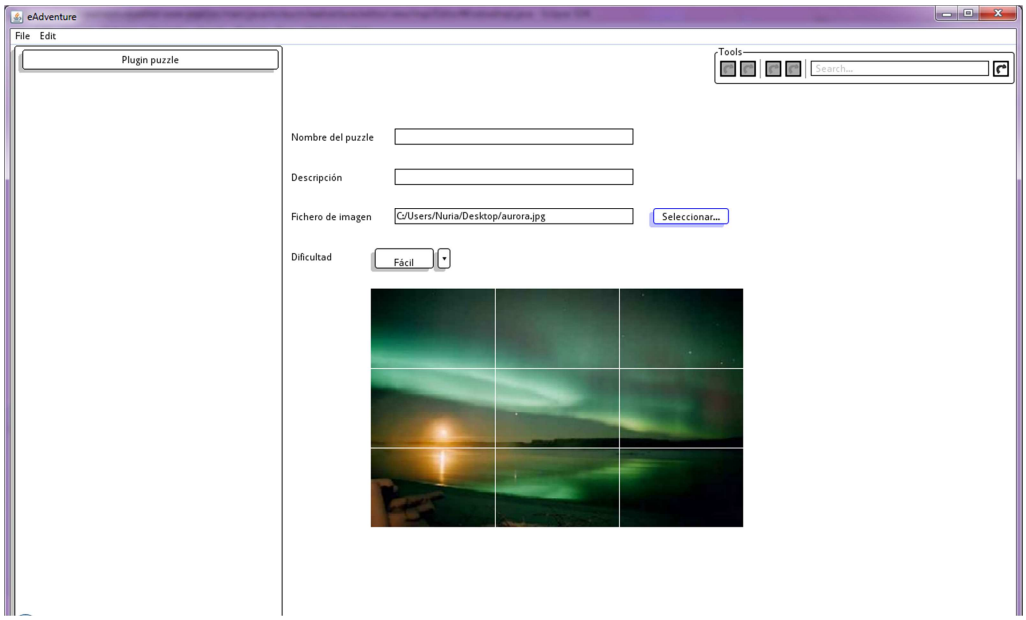

Figura 66. Prototipo editor plugin puzle

Figura 67. Prototipo motor plugin puzle

Se puede observar en la imagen los campos para nombre, descripción, fichero de imagen, así como selección de nivel de dificultad y otros posibles parámetros como son un contador de tiempo, o la opción de rendición. También se puede ver que una vez el usuario indica la imagen que quiere cargar a través del campo 'Fichero de imagen', se muestra una vista previa de cómo quedaría dicha imagen con sus divisiones, dependiendo del grado de dificultad seleccionado en ese momento.

En el desarrollo de la parte del motor se va a necesitar tener un registro de una lista de casillas para identificar cada pieza con su id y url de la imagen asociada. Durante la ejecución, en cada movimiento se intercambia la posición de una pieza y el hueco, comprobando que el desplazamiento se puede hacer, y en caso afirmativo, tras llevarlo a cabo, verificando si el puzle se ha ordenado o no. Si se ha conseguido ordenar se termina el juego, y en caso contrario, el rompecabezas se repinta con las nuevas posiciones de las casillas y se continua jugando. En su inicialización se va a recibir la imagen seleccionada en la interfaz de editor, así como los distintos datos que identifican ese puzle.

Implementación

En la parte del editor hemos dividido el desarrollo de la interfaz en dos partes, por un lado la parte de las interfaces, y por otra la implementación de esas interfaces cada una en su correspondiente paquete dentro de la estructura de proyectos de e-Adventure.

Hemos seguido los convenios de nombres y estructura de paquetes que contempla E-Adventure. De esta forma, contamos con interfaces y clases como EAdPuzzle y EAdPuzzleImpl que contienen la estructura para que los datos que se configuran en la interfaz sea almacenados correctamente en el XML a través de la clase EAdPuzzleControl, donde se almacenará el juego y

que de esta forma se pueda interpretar correctamente desde el motor cuando se ejecute el juego. La clase EAdPuzzlePlugin, se encarga de la inicialización del plugin en el editor, es decir, de cargar el panel y la correspondiente pestaña dentro de la ventana de e-Adventure.

Por otro lado, en la parte del motor, se han desarrollado clases que gestionan el puzle en el motor: NPuzzle gestiona los movimientos y restricciones del puzle en el juego y StartNPuzzleEffect se encarga de ordenar de forma aleatoria las piezas del puzle cuando se pulsa el botón de "Iniciar" el juego.

Algunas capturas de pantalla de la visión real que ha tomado el plugin respecto al editor se pueden observar en la figura anterior (Figura 66).

En el panel izquierdo se muestra un botón que habilita el plugin al pulsarlo. Se observa un ejemplo de puzle a partir de una imagen con un grado de dificultad 'Fácil', y por lo tanto, dividido en una matriz 3 x 3.

Dificultades encontradas

Las principales dificultades con las que nos hemos cruzado se han derivado del desconocimiento de la implementación de la plataforma e-Adventure. Ha sido necesario realizar un estudio del código para saber la forma en la que debíamos trabajar a la hora de hacer este primer plugin. Había que conocer la estructura del proyecto y su funcionamiento, así como los distintos módulos que componen e-Adventure.

La tarea más complicada se podría decir que ha sido el conseguir una integración satisfactoria del funcionamiento de nuestro plugin en la herramienta e-Adventure, sobre todo, en el aspecto del motor. La parte del editor nos ha resultado mucho más fácil al tratarse de una tarea más sencilla.

Otro problema, es el caso de que sea una herramienta en desarrollo con continuos avances y modificaciones, lo que nos supone un mayor esfuerzo a la hora de analizar su estructura. Además, como todavía no está finalizada, nos hemos topado con recursos que no han sido desarrollados todavía, y de los que hemos necesitado echar mano para el funcionamiento de esta extensión.

Conclusiones

El trabajo en este plugin nos ha permitido obtener un conocimiento más profundo del desarrollo de la plataforma e-Adventure y de las distintas herramientas que se emplean para ello.

Nos ha costado más tiempo del esperado implementarlo, pero al tratarse del primero creemos que es algo lógico, puesto que era nuestra toma de contacto con la aplicación. A partir de este plugin, los siguientes tendrán en principio un coste de tiempo mucho menor, ya que evitamos el tiempo extra necesario para entender y analizar el código.

En lo referente al plugin del puzle, creemos que se trataría de una extensión con mucha utilidad para la finalidad educativa de la herramienta, puesto que fomenta la agilidad y destreza espacial del jugador. Se trata además, de un juego muy conocido y popular, del que todo el mundo conoce el funcionamiento y ha jugado alguna vez en su vida. Ofrece la posibilidad de incluir cualquier tipo de imagen en el puzle, lo que aumenta su objetivo de

aprendizaje, al poder establecer capturas del tema educativo que se quiera enseñar con un determinado juego.

6.2.2 Plugin del puzle arrastrando piezas

Descripción

Este plugin es una nueva versión sobre el plugin del puzle anterior.

Se trata de un plugin basado en un puzle al estilo de los puzles de mesa que todos conocemos. Se podrá diseñar un rompecabezas en el que, a partir de una imagen seleccionada, esta se dividirá en diferentes piezas o fichas rectangulares, que serán descolocadas. Tras esto, el jugador deberá arrastrar cada una de las piezas a su posición original.

Se muestran imágenes de este tipo de juegos en las figuras (Figura 68, Figura 69 y Figura 70).

Alcance

Al igual que nuestro plugin anterior, este se divide en parte de editor y parte para motor. En lo referente a editor se proporciona un panel, que se acopla a la interfaz principal de e-Adventure, en el cual se facilitan las distintas opciones para que el usuario pueda crear un puzle según sus preferencias. Esto es, en el panel de creación del puzle, se podrá elegir la imagen que se desea usar como fondo, la dificultad que se desee, etc.

Figura 68. Imagen de puzle[53]

Con respecto al motor, se han desarrollado las funciones necesarias para poder desplazar las piezas mediante el sistema “drag and drop”, es decir, arrastrando y soltando cada una de las piezas hasta la posición final.

El jugador observará una ventana con las piezas ya divididas y desordenadas y tendrá que arrastrar cada una de las piezas a su posición correcta, siguiendo una rejilla base sobre la que deben situarse las piezas.

53 Imagen de puzle http://elmundodemarisol.blogspot.com

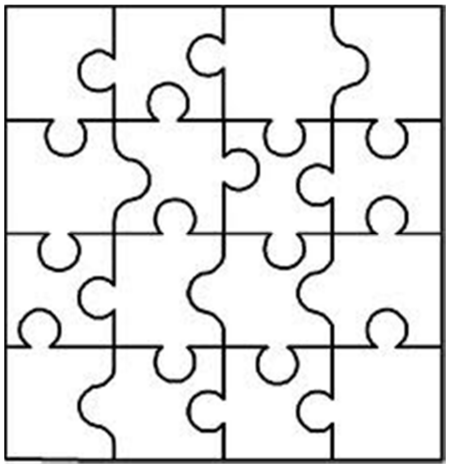

Figura 69. Imagen de puzle[54]

Figura 70. Imagen de puzle[55]

Requisitos

Esta extensión debe permitir generar puzles a partir de una imagen dada, la cual se dividirá en diferentes porciones que se desordenarán y el usuario deberá volver a ordenar adecuadamente.

[54] Imagen de puzle http://www.dreamlandmagic.net/halloween/fun.html
[55] Imagen de puzle http://www.adrada.es/psicomotricidad/36-72%20meses.html

Para generar los puzles se cuenta con:

- Panel que se adhiera a la interfaz gráfica principal y que contenga campos de texto para introducir datos representativos para cada rompecabezas, como pueden ser su nombre, descripción, url de la imagen, dificultad, etc.
- Posibilidad de seleccionar la imagen del puzle mediante una ventana de diálogo que permita elegir los archivos.
- Selección de distintos niveles de dificultad para indicar la división que haya que hacer de filas por columnas para la imagen.

Respecto a la interacción del usuario cuando se está ejecutando este sistema:

- Visualización de las piezas formando un conjunto de imagenes desordenadas de forma aleatoria.
- Visualización de una rejilla base sobre la que se irán colocando las piezas una vez que se hayan movido al lugar correcto para cada una de ellas.
- Posible visualización de la imagen original como pista.
- Desplazamiento de las piezas utilizando el sistema de "drag and drop".
- Finalización del juego cuando el usuario consiga formar el puzle correctamente, llegando al dibujo original, o bien, se termina tras alcanzar un tiempo límite de juego; aspecto sin determinar, de momento no existe contador de tiempo y consideramos que se debe resolver el rompecabezas para darlo por finalizado.

Diseño del plugin

Entre las ideas generales en el diseño de este plugin podemos distinguir las siguientes:

- Tal y como se ha comentado anteriormente se va a establecer el movimiento a las casillas con el sistema de arrastrado.
- A nivel de usuario, durante la ejecución del sistema, su intención será mover las casillas del puzle, arrastrándolas hacia la posición destino que considere correcta.
- El número de piezas posibles para dividir la imagen, relativo al grado de dificultad, se establece en tres configuraciones distintas:

    - Nivel fácil: división en 3 x 3
    - Nivel medio: división en 4 x 4
    - Nivel difícil: división en 5 x 5

El editor se va a caracterizar por disponer de una interfaz gráfica con las características descritas en la sección de requisitos. Se facilita un prototipo que muestra el diseño de dicha interfaz, que coincide con la del puzle original (Figura 66).

Se puede observar en la imagen los campos para nombre, descripción, fichero de imagen, así como selección de nivel de dificultad y otros posibles

parámetros como son un contador de tiempo, o la opción de rendición. También se puede ver que una vez el usuario indica la imagen que quiere cargar a través del campo 'Fichero de imagen', se muestra una vista previa de cómo quedaría dicha imagen con sus divisiones, dependiendo del grado de dificultad seleccionado en ese momento.

En el desarrollo de la parte del motor se va a necesitar tener un registro de una lista de casillas para identificar cada pieza con su id y url de la imagen asociada. Durante la ejecución, en cada movimiento se arrastra una pieza a un hueco de la rejilla base. Si el movimiento se puede hacer, es la casilla destino de la pieza, se verifica si están todas las piezas en su posición original, concluyendo el juego, o no, repitiendo el proceso de nuevo. En su inicialización se va a recibir la imagen seleccionada en la interfaz de editor, así como los distintos datos que identifican ese puzle.

Implementación

Para el desarrollo del plugin se ha pensado que, como la interfaz de usuario para la parte del editor corresponde con la del plugin de puzle anterior, reutilizarla para este nuevo plugin usando las dependencias que el framework “jin-plugin” provee.

Al igual que con el plugin anterior, hemos seguido el convenio de nombres y estructura de paquetes de EAdventure.

Además, hemos reutilizado tanto clases como interfaces del puzle anterior como EAdPuzzle, EAdPuzzleImpl y EAdPieza que gestionan los datos necesarios para identificar cada puzle y cada pieza respectivamente. También la clase EAdPuzzleView, que se encarga de la realización de la interfaz gráfica del plugin para el editor.

Para la parte del motor hemos tenido que crear una clase PuzzleBasicoGO contiene los métodos que resuelven los algoritmos de desplazamiento de casillas y todo el grueso de gestión de la ejecución del juego.

Algunas capturas de pantalla de la visión real que ha tomado el plugin respecto al editor se pueden observar en la figura anterior (Figura 66). En el panel izquierdo se muestra un botón que habilita el plugin al pulsarlo. Se observa un ejemplo de puzle a partir de una imagen con un grado de dificultad 'Fácil', y por lo tanto, dividido en una matriz 3 x 3.

Dificultades encontradas

Una de las dificultades con las que nos hemos encontrado para este segundo plugin han sido básicamente las que nos provocaron el entender el uso de dependencias en el framework de plugins, si bien es cierto, que no fue excesivamente complejo.

Asimismo, nos ha provocado bastantes dificultades la lógica del plugin en cuanto al método de movimientos de las piezas mediante el uso de arrastrar y soltar.

Al igual que con el primer plugin, nuestro principal problema es entender el correcto funcionamiento de e-Adventure y de Google Guice, ya que, aún nos costaba bastante entender cómo actúa este.

Conclusiones

Hemos seguido teniendo problemas por el uso de la herramienta e-Adventure porque no habíamos trabajado antes en un ambiente de trabajo tan cambiante, ya que las prácticas que habíamos realizado durante la carrera estaban basadas sobre herramientas ya acabadas y probadas. Aun así, estamos poco a poco acostumbrándonos a este método de trabajo y lo consideramos una experiencia muy positiva puesto que, creemos, se asemeja a un ambiente de trabajo más real.

Al desarrollar este plugin, nos hemos dado cuenta que el tema de las dependencias en los plugins para e-Adventure no nos va a resultar excesivamente complicado.

En lo referente al plugin del puzle en sí, debemos comentar que se trataría de un minijuego con un gran potencial educativo, sobre todo a nivel visual y de memoria. El hecho de poder desplazar las piezas mediante el arrastrado de las mismas consideramos que aporta un toque ameno al aprendizaje.

6.2.3 Plugin de las preguntas/trivial

Descripción

Se trata de un plugin que encapsula el típico juego de preguntas y respuestas. Se establecen un conjunto de preguntas, las cuales tienen múltiples opciones de respuesta. Por ejemplo, se plantea una cuestión con cuatro posibles soluciones, de las cuales solo una de ellas será correcta. Pueden incluirse preguntas de todo tipo, o bien centrarse en un tema concreto, un ámbito académico determinado, etc. Es decir, es totalmente a gusto de la persona que diseña el conjunto de preguntas con sus respuestas asociadas.

Se puede comprobar con las imágenes facilitadas de ejemplo (figuras: Figura 71, Figura 72 y Figura 73) que existen multitud de variantes y aplicaciones desarrolladas para este tipo de juego.

Alcance

De la misma forma que los plugins anteriores, se distingue para esta extensión la parte de editor y la parte para motor.

En el editor se le va a permitir al usuario definir un conjunto de preguntas, junto a las respuestas que asocie para cada una. Una de estas respuestas se indicará como la correcta, y el resto serán erróneas. La persona que diseña el juego podrá incluir el número de preguntas que desee.

Figura 71. Ejemplo de trivial del juego Buzz[56]

En el motor se le irán mostrando al jugador el grupo de cuestiones que conforman el juego, y tendrá que ir respondiéndolas en orden. Deberá elegir la respuesta correcta pulsando el botón correspondiente a dicha solución. Así, sucesivamente, se irá avanzando en las posteriores preguntas, hasta terminar este minijuego.

Figura 72. Ejemplo[57] *de trivial 2*

56 Ejemplo de trivial del juego Buzz http://consolas.com/videojuegos/buzz-el-multiconcurso-49407
57 Ejemplo de trivial 2 http://viciogeek.com/juego-de-preguntas-y-respuestas

Requisitos

Plugin para la elaboración de grupos de preguntas con respuestas asociadas, de las cuales sólo una es la correcta.

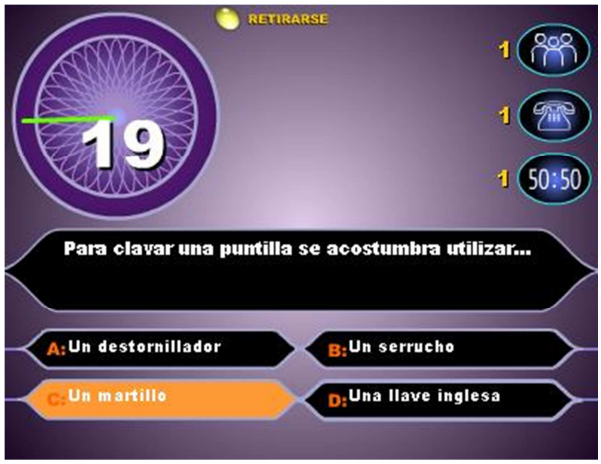

Figura 73. Ejemplo[58] de trivial 3

En el diseño del juego se cuenta con:

- Una tabla para ir introduciendo el conjunto pregunta-respuestas del minijuego.
- Cada fila de la tabla representará las distintas preguntas de juego, y después se tendrá una columna para redactar la pregunta en concreto, y sucesivas columnas para indicar la respuesta correcta, y las erróneas.
- Campos para incluir nombre del conjunto de preguntas, instrucciones de juego, etc.
- Botones para añadir nuevas preguntas, eliminar alguna ya introducida, etc.
- Posibilidad de elegir imágenes como respuestas.

Para la interacción con el juego:

- Aleatoriedad en el orden de visualización de las respuestas asociadas a una pregunta.
- Se le mostrará al usuario las opciones de respuesta posible, de las que deberá seleccionar sólo aquella que considere correcta.
- Tratamiento de las respuestas como zonas interactivas, permitiendo que sean pulsadas por el jugador mediante el ratón.
- Una vez seleccionada la respuesta se procederá a mostrar la siguiente pregunta de la colección.

58 Ejemplo de trivial 3 http://infodescargas.org/el-juego-%C2%BFquien-quiere-ser-millonario-para-descarga

Diseño del plugin

Entre las ideas generales en el diseño de este plugin podemos distinguir las siguientes:

- Como se ha comentado, el plugin mostrará una pregunta introducida y un conjunto de respuestas posibles de las que sólo una será correcta.
- A nivel de editor, el usuario dispondrá de una interfaz en la que se podrá introducir una pregunta y tantas respuestas como desee, marcando la que es la correcta. También se elegirá el nombre del conjunto de preguntas y el orden en el que se generarán cada una de ellas. Además, se podrán elegir imágenes como respuestas a las preguntas.
- En tiempo de ejecución, el usuario visualizará una pantalla con la pregunta y respuestas introducidas. El usuario tan sólo tendrá que "clickar" sobre una de estas respuestas para obtener el resultado. Una respuesta negativa generará un mensaje de error y/o terminará la ejecución del plugin.

El editor se va a caracterizar por disponer de una interfaz gráfica con las características descritas en la sección de requisitos. Se facilita un prototipo que muestra el diseño de dicha interfaz (Figura 74):

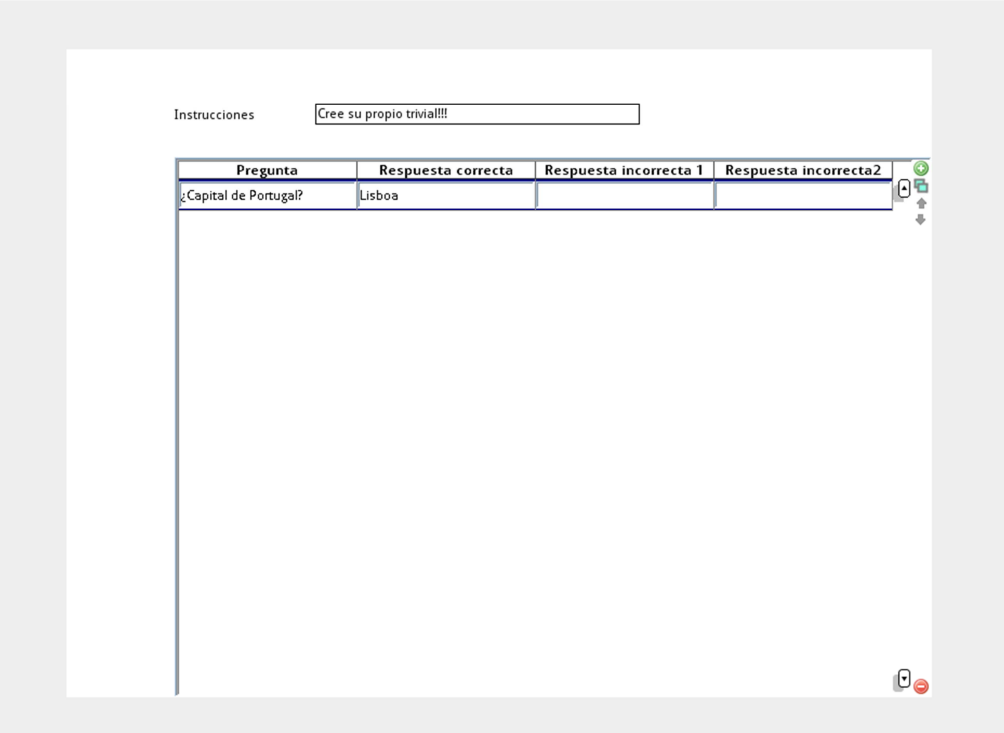

Figura 74. Imagen del editor

Se puede observar en la imagen los campos para la pregunta y cada una de las respuestas. En la parte del motor, se necesitará tan sólo comprobar si la respuesta seleccionada es la correcta o una de las incorrectas y actuar de

un modo concreto dependiendo de esto. Para ello, necesitará una variable que contenga la respuesta correcta y la lista de acciones que generará el plugin en caso de elegir la respuesta correcta o una de las incorrectas.

Implementación

Para el desarrollo del plugin se han seguido los convenios de nombres y estructura de paquetes que contempla EAdventure. De esta forma, contamos con interfaces y clases como EAdQuestion, EAdQuestionControl, EAdQuestionImpl y EAdQuestionPlugin.

Algunas capturas de pantalla de la visión real que ha tomado el plugin respecto al editor (Figura 75 y Figura 76).

Figura 75. Ejemplo de ejecución 1 del trivial

Figura 76. Ejemplo de ejecución 2 del trivial

Dificultades encontradas

Las principales dificultades con las que nos hemos encontrado han surgido a raíz del uso de un gestor de texto para e-Adventure. Al final nos hemos decantado por usar una modificación del que provee e-Adventure por defecto, añadiendo la posibilidad de añadir imágenes en vez de texto al mismo.

Tuvimos, asimismo, grandes dificultades con la manera de colocar en pantalla las preguntas, dependiendo de si se trataba de una pregunta cuya respuesta contenía sólo texto o imágenes, pues estas últimas pueden tener tamaño variable, forma, etc y había que contemplar todos los casos.

Conclusiones

Como suponíamos, este plugin nos llevó menos tiempo que el primero desarrollado puesto que nuestra dificultad con respecto al conocimiento de la plataforma base ya era mucho mayor. En cualquier caso y, pese a las dificultades, creemos que el resultado final es satisfactorio.

Además, este plugin podría incluir mejoras tales como un contador de tiempo, tener un conjunto de preguntas y respuestas predefinidas, etc. Aparte de esto, creemos que sería factible e interesante adaptar el plugin de tal manera que se pudieran "desacoplar" las preguntas entre sí y así hacer posible que fueran apareciendo cada una de ellas a lo largo del juego en el que esté incluido, en vez de ejecutarse todas seguidas.

6.2.4 Plugin memorión

Descripción

El plugin de nombre 'Memorión' consiste en un juego cuyo objetivo es el desarrollo y ejercicio de la memoria del jugador. El juego presenta un conjunto de imágenes, relacionadas por parejas, que se muestran durante unos segundos al inicio. Seguidamente, se ocultarán estas imágenes para que el usuario vaya seleccionando primero una y después la que considera que es su pareja. Por ello, como se ha comentado, hay que memorizar y acordarse de la situación de las distintas instantáneas.

Se muestran capturas de este tipo de juegos (Figura 77, Figura 78 y Figura 79).

Figura 77. Memorión[59]

Figura 78. Ejemplo 2 de memorión[60]

Figura 79. Ejemplo 3 de memorión[61]

59 Ejemplo de memorión http://unmundolibre.net
60 Ejemplo 2 de memorión http://gabicuesta.blogspot.com
61 Ejemplo 3 de memorión http://www.taringa.net

Alcance

Para este plugin, en la parte de editor se establecen los correspondientes campos para nombre, descripción, etc…, de forma análoga a los anteriores. Después se dispone de una tabla en la que el creador del juego podrá ir introduciendo las imágenes que desea incluir. Tan solo deberá añadir una vez la imagen que quiere comparar con su pareja, puesto que el motor posteriormente, ya se encargará de duplicar las imágenes para poder jugar. Además se ofrece la posibilidad de visualizar las imágenes añadidas, teniendo seleccionada la que se quiera ver y pulsando a continuación el botón correspondiente.

El motor, al iniciarse la partida, se encargará de duplicar las figuras y distribuirlas aleatoriamente en la pantalla. Se le mostrarán al jugador dadas la vuelta para que vaya pinchando sobre ellas, y de esta manera ir asociando las parejas.

Requisitos

Se trata de realizar un plugin para la asociación de parejas mediante el recuerdo de la posición que ocupan.

El diseño del juego cuenta con:

- Campos de texto para introducir nombre del juego y descripción.
- Tabla para poder añadir las figuras que se tendrán que asociar.
- Interacción con la tabla para poder añadir, eliminar o modificar las imágenes.
- Botón que permita obtener una vista previa de una imagen seleccionada del grupo de las que se hayan incluido.

En la ejecución del juego destacar:

- Duplicación por parte del motor de cada imagen incorporada.
- Aleatoriedad en las posiciones de las figuras.
- Imágenes dadas la vuelta en un principio del juego.
- El jugador irá pinchando primero una imagen y seguido la segunda que considere que es la asociada. Al pinchar se le irán mostrando.
- Si se acierta en la asociación de imágenes, éstas se quedarán descubiertas. En caso de fallo, volverán a darse la vuelta.

Diseño del plugin

El diseño de este plugin sigue las líneas marcadas por el alcance y los requisitos expuestos en los apartados anteriores.

A continuación, se puede ver una figura de este diseño (Figura 80).

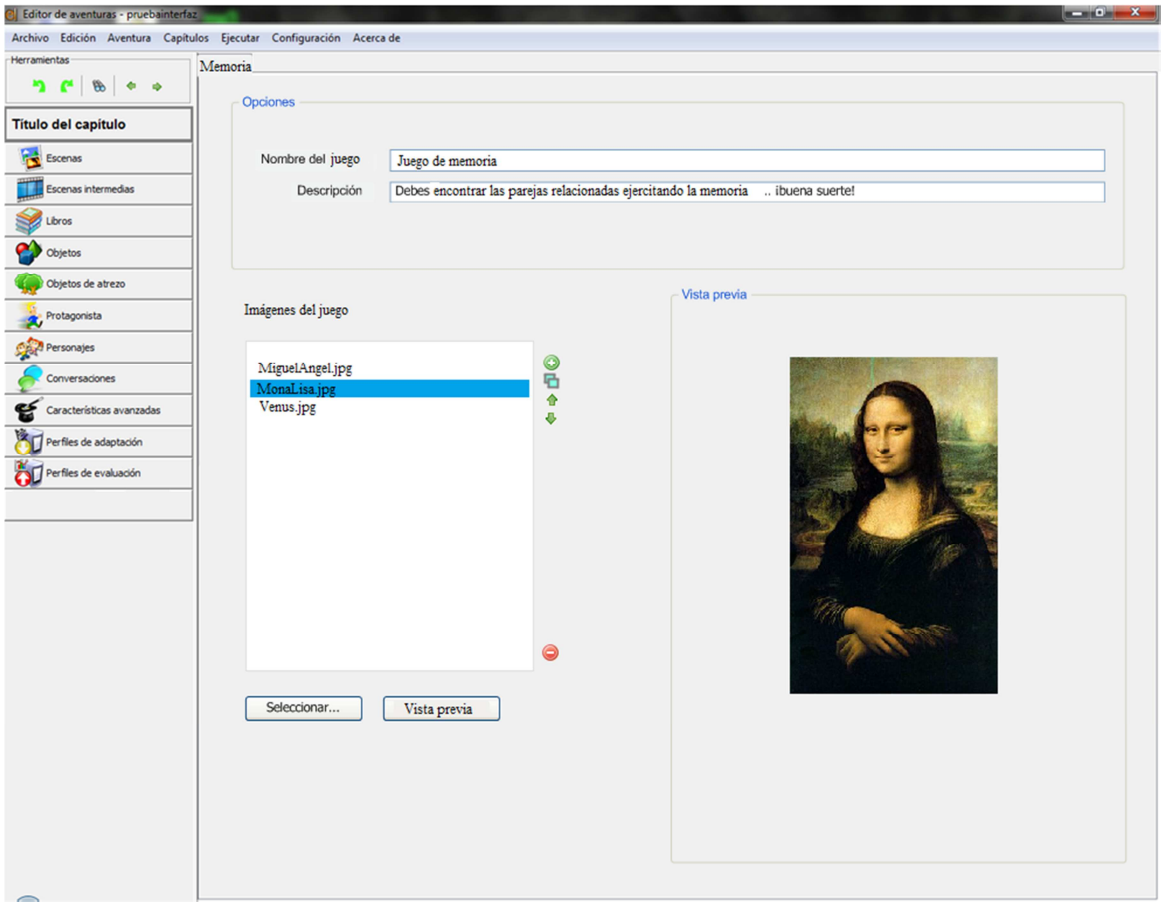

Figura 80. Prototipo editor plugin memorión

Implementación

En el desarrollo de este plugin, de la misma forma que en los otros, se han seguido sucesivas fases para la implementación del motor. Se realizó una primera versión operativa y con funcionamiento correcto del juego siguiendo la estructura de efectos en e-Adventure, creándonos los propios efectos necesarios para el juego y estableciendo el renderizado y actualización del juego de forma explícita.

Posteriormente, nos aconsejaron que simplificáramos todo lo que pudiésemos el código y nos sirviéramos de los efectos ya existentes en e-Adventure. De esta forma, y empleando los eventos, efectos y condiciones que ofrece la herramienta, nos era suficiente para lograr desarrollar este tipo de plugins.

Se presentan capturas de pantalla del resultado final en la parte de diseño, y en la ejecución de la partida (Figura 81 y Figura 82).

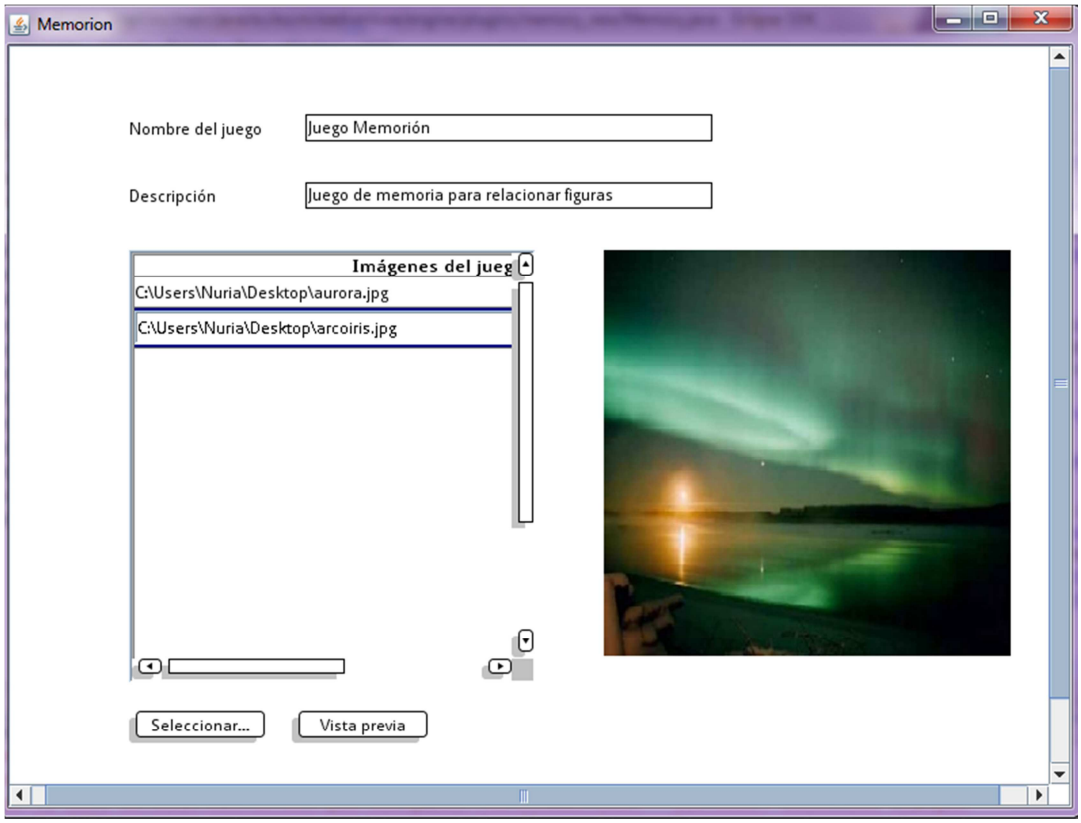

Figura 81. Captura editor plugin memorión

Dificultades encontradas

Una vez desarrollados los otros plugins, la primera versión del motor no supuso ningún problema y se realizó de forma rápida. No nos costó mucho conseguir que este juego funcionara correctamente tras haber aprendido el funcionamiento de la herramienta con los anteriores.

Para la segunda versión los principales problemas estuvieron en la forma de establecer las condiciones de la partida. La incorporación de los efectos necesarios para su ejecución resultó sencilla, pero la adecuación de eventos y condiciones nos presentó algunas dificultades a la hora de lograr un buen funcionamiento del juego.

Conclusiones

El juego del memorión tiene una gran faceta educativa, puesto que ayuda a agilizar la memoria, y a la vez con sus imágenes se puede aprender de cualquier ámbito académico. Por lo tanto, consideramos que es un caso de estudio perfecto para una herramienta como e-Adventure, centrada en los juegos educativos.

Figura 82. Ejemplo de ejecución plugin memorión

Su desarrollo ha necesitado de menor tiempo debido al aprendizaje que hemos ido adquiriendo en la realización de los distintos plugins.

Capítulo VII
Conclusiones

Tabla de contenidos

7. Introducción

Este capítulo se va a centrar en la evaluación de la consecución o no de los objetivos planteados al inicio del proyecto.

7.1 Objetivo 1

Antes de nada, recordemos en qué consistía exactamente el primero de los objetivos:

Producción de un sistema de extensiones para la herramienta de desarrollo e-Adventure que permita implementar funcionalidades extra que puedan ser incorporadas de manera independiente al núcleo principal.

Con respecto a este primer objetivo podemos asegurar que se ha cumplido completamente.

Se ha desarrollado un sistema de extensiones para e-Adventure por el cual se pueden acoplar las funcionalidades que deseemos y de la manera más sencilla posible de entre las distintas opciones que se analizaron en un principio.

Creemos que esto contribuye a la mejora de la herramienta, haciéndola más flexible.

7.2 Objetivo 2

El objetivo 2 consistía en lo siguiente:

Desarrollo de plugins de ejemplo para probar la funcionalidad del framework.

Como hemos podido observar a lo largo de esta memoria, se han realizado los plugins necesarios para comprobar que nuestro sistema funciona correctamente.

Al realizar estos plugins hemos ido observando los errores que surgían, situándonos en la perspectiva de un desarrollador ajeno al núcleo de e-Adventure, y lo hemos ido solventando, teniendo en cuenta la facilidad de uso inferida del objetivo 1.

7.3 Conclusiones finales

Bajo nuestro punto de vista, se han conseguido completamente los dos objetivos principales del proyecto.

Para conseguirlos hemos seguido una serie de pasos que considerábamos fundamentales si queríamos cumplirlos.

Primero, realizamos un análisis de la plataforma para comprobar la estructura de la misma. Esto fue necesario para saber la forma en la que un sistema de plugins podría ser acoplado.

A continuación se realizó un amplio estudio de herramientas para la creación de videojuegos, fijándonos principalmente en aquellas que disponían de un sistema de plugins, y asimilando las diferentes opciones que cada una de ellas aportaba a su herramienta. Además, también se realizó un exhaustivo examen de las principales plataformas para la gestión de plugins, comparando sus ventajas y desventajas y comprobando las diferentes características de cada una de ellas.

Tras esto, se adaptó el núcleo principal de e-Adventure para poder insertar el gestor de extensiones ideado. Más tarde, se desarrolló el sistema de plugins, basándonos en el gestor Jin-Plugin, que había sido elegido por su sencillez y poco peso como mayores ventajas. Una vez realizado el sistema de plugins, se desarrollaron una serie de extensiones de prueba que pudieran confirmar el correcto funcionamiento del framework. Con cada uno de los plugins creados se fue modificando la herramienta original para poder adaptarla a la plataforma e-Adventure, obteniendo finalmente, un sistema estable y acorde a lo establecido en los objetivos 1 y 2.

Bibliografía

8. Bibliografía y referencias

[1] Davison, A. (2005). Killer Game Programming in Java. United States of America: O'Reilly Media, Inc.

[2] http://e-adventure.e-ucm.es <e-Adventure> web site.

[3] Anuario 2009 aDeSe (Asociación Española de distribuidores y Editores de Software de Entretenimiento)
http://www.adese.es/pdf/Anuario2009aDeSe.pdf

[4] IDATE: Serious Games - A 10 Billion Euro Market In 2015 Serious Games Market.
http://seriousgamesmarket.blogspot.com/2010/08/idate-serious-games-10-billion-euro.html

[5] Gee, J.P. (2003). What videogames have to teach us about learning and literacy. New York: Palgrave McMillan.

[6] Ondrejka, C., Cook, J., Conklin, M. (2005). How user content changes everything. Games Learning and Society Conference. June 23, 2005.

[7] Aldrich, C. (2004). Simulations and the Future of Learning: an Innovative (and Perhaps Revolutionary) Approach to e-Learning. San Francisco, CA: Pfeiffer.

[8] Aldrich, C. (2005). Learning by Doing: A Comprehensive Guide to Simulations, Computer Games and Pedagogy in e-Learning and Other Educational Experiences. San Francisco, CA: Pfeiffer.

[9] Página del autor de Jin-Plugin: http://code.google.com/p/jin-plugin

[10] Proyecto Google Guice http://code.google.com/p/google-guice/

[11] Página oficial de JUnit http://www.junit.org/

[12] Torrente Vigil, J., Del Blanco, A., J. Marchiori, E. La plataforma <e-Adventure> Manual de usuario. Universidad Complutense de Madrid.

[13] Malone, T. (1982). What makes computer games fun? SIGSOC Bulletin, 13(2-3), 143.

[14] Prensky, M. (2001). Digital Game Based Learning. New York: McGraw- Hill.

[15] Michael, D. and Chen, S. (2006). Serious Games: Games that Educate, Train, and Inform. Boston, MA: Thomson.

[16] Pressman, R. S. (2005). Ingeniería del Software. McGraw-Hill Interamericana.